MacTech Magazine
Vol. 14, No. 2 • February 1998

For Macintosh Programmers & Developers

INSIDE:
The Eddy Awards

MacTech®

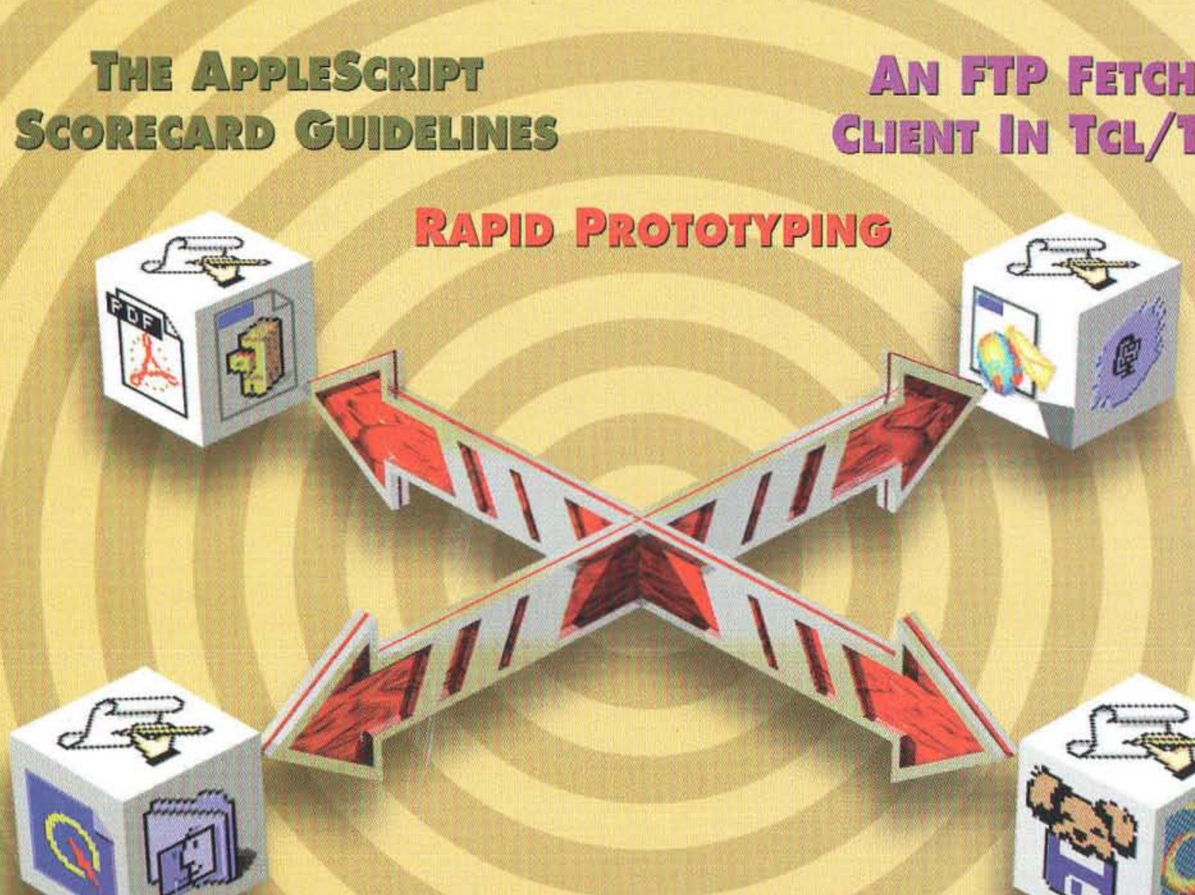
Includes Special develop Section by Apple Computer, Inc.

Scripting

The AppleScript Scorecard Guidelines

An FTP Fetch Client in Tcl/Tk

Rapid Prototyping



Plus:

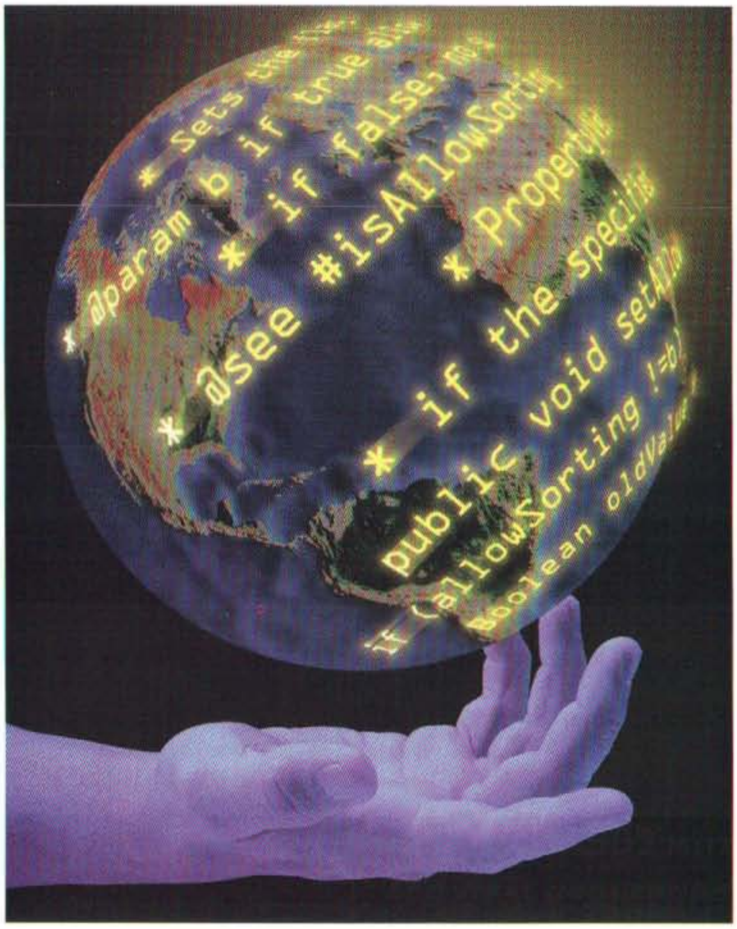
Contextual Menu Modules

MacPerl:
A Developer's Overview

5.85 US
6.95 Canada
SSN 1067-8360
rinted in U.S.A.

• cafe.symantec.com • cafe.symantec.com • cafe.symantec.com • cafe.symantec.com • cafe.symantec.com •

Visual Café™ for Java™: development tools so hot, the competition can't touch them.



	Symantec's Visual Café for Java*	MetroWerks Code Warrior Pro**
Complete RAD Java IDE	Yes	No
Form Designer	Yes	Yes
Display JavaBeans™ in Form Designer	Yes	No
Macintosh runtime for Java (MRJ 2.0 with JDK1.1)	Yes	Yes
Easy conversion from JDK 1.02 to JDK 1.1	Yes	No
Over 100+ JavaBeans with JavaBean Creator Assistant	Yes	No
Interaction Editor	Yes	No
Autocode generation (work the way you want to work)	Yes	No
Database connectivity and support for Filemaker Pro with BlueWorld Lasso	Yes	No
HTML Web authoring tool (Visual Page)	Yes	No
Native support for Oracle, Informix, Sybase MS SQL, & 40+databases via ODBC	Yes	No

*All features included in the Database Development Edition **Features as of October 15,1997

The most powerful Java and JavaBeans environment.

The most powerful Java development solution for databases.

For Macintosh

Download a free trial copy
http://cafe.symantec.com

1-800-971-1582 MT121

The next generation of Java Tools

SYMANTEC.™

JAVA MacOS

• cafe.symantec.com • cafe.symantec.com • cafe.symantec.com • cafe.symantec.com • cafe.symantec.com •

Do it your way!

No application, no matter how many features it includes, can do everything. People and organizations spend thousands of dollars buying and upgrading software hoping to end up with a package that fills all their needs.

There is an easier way.

FaceSpan™ 3.0 is a cutting edge interface design and rapid application development (RAD) tool which gives you the power to build and customize Macintosh applications quickly and easily.

Whether you want to create a simple floating tool palette that gives you quick access to your favorite scripts, have an idea for a new shareware program or need to tie together the functionality of multiple applications...

Automate often-repeated tasks.

Rapidly build applications & prototypes.

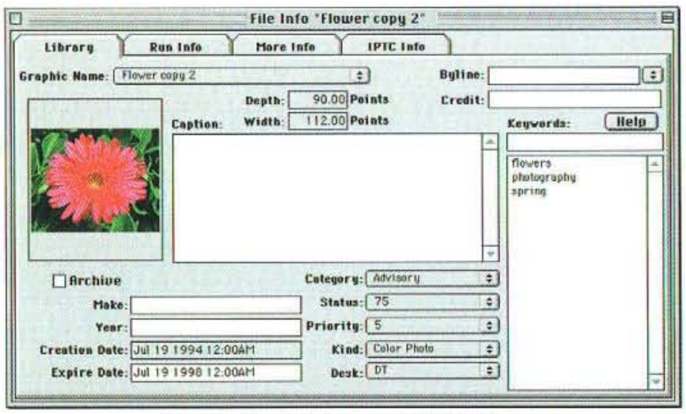
Customize existing applications.

Integrate multiple applications.

Personalize your computing environment.

FaceSpan 3.0

FaceSpan is the answer.


File Info "Flower copy 2"

Library | Run Info | More Info | IPTC Info

Graphic Name: Flower copy 2
Byline:
Credit:
Depth: 90.00 Points
Width: 112.00 Points
Caption:
Keywords: Help
flowers
photography
spring

☐ Archive
Make:
Year:
Creation Date: Jul 19 1994 12:00AM
Expire Date: Jul 19 1998 12:00AM
Category: Advisory
Status: 75
Priority: 5
Kind: Color Photo
Desk: DT

Example of an interface created in FaceSpan.

Special Introductory Price: $149

FOR ADDITIONAL INFORMATION AND UPGRADE PRICING:

1-800-322-3772 (USA)
1-801-226-2984 (Other Countries)
1-801-226-8438 (Fax)
facespan@dtint.com (E-mail)

http://www.facespan.com

Features new to FaceSpan 3.0 allow you to:

New! Expanded tool palette!

- Create interfaces and applications that conform to the Mac OS 8 look and feel.
- Utilize new supported display objects such as tab panels, disclosure triangles and bevel buttons.
- Define linkages that hide/show window items when another item is hilited.
- Develop hierarchical menus and assign command key modifiers directly.
- Edit all your project scripts with increased support for external editors such as Scripter™ and Script Debugger™.
- Launch FaceSpan applications up to 5X faster.

FaceSpan™
A Product of Digital Technology International

For Macintosh Programmers & Developers

MacTech® MAGAZINE

A publication of XPLAIN CORPORATION

How To Communicate With Us

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail?**

If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

If you would like a subscription or need customer service, feel free to contact Developer Depot Customer Service at 800-MACDEV-1

DEPARTMENTS	E-Mail/URL
Orders, Circulation, & Customer Service	cust_service@devdepot.com
Press Releases	press_releases@mactech.com
Ad Sales	ad_sales@mactech.com
Editorial	editorial@mactech.com
Programmer's Challenge	prog_challenge@mactech.com
Online Support	online@mactech.com
Accounting	accounting@mactech.com
Marketing	marketing@mactech.com
General	info@mactech.com
Web Site (articles, info, URLs and more...)	http://www.mactech.com

MacTech Magazine

MacTech Magazine is grateful to the following individuals who contribute on a regular basis. We encourage others to share the technology.

We are dedicated to the distribution of useful programming information without regard to Apple's developer status.

For information on submitting articles, ask us for our ***writer's kit*** *which includes the terms and conditions upon which we publish articles.*

Editorial Board of Advisors

Scott T Boyd, Jordan J. Mattson, Jim Straus, and Jon Wiederspan

Editors

Publisher • Neil Ticktin
Editor-in-Chief • Eric Gundrum
Managing Editor • Jessica Courtney
Online Support • Nick "nick.c" DeMello, Kevin Avila

Contributing and Technical Editors

Java • Gary Little, Apple Computer, Inc.
Components • Tantek Çelik, Microsoft Corporation
Internet • Carl de Cordova, Apple Computer, Inc., Jeff Ganyard
MacDev-1™ • Rich Siegel, Bare Bones Software, Inc.
Mac OS 8 • Steve Kiene, Mindvision
MagicCap/TeleScript • Richard Clark
Performance Programming • Jim Gochee, Connectix
Product Reviews • Ed Ringel
Rhapsody • Michael Rutman

Regular Columnists

Getting Started • Dave Mark
Programmer's Challenge • Bob Boonstra
From the Factory Floor • Dave Mark, Metrowerks
Tips & Tidbits • Steve Sisak
MacTech Online • Nick "nick.c" DeMello

Xplain Corporation

Chief Executive Officer • Neil Ticktin
Chief Operating Officer • Andrea J. Sniderman
Controller • Heather Principe
Advertising Executive • Fran Stern
MarComm Manager • Susan M. Whitney
Events Manager • Susan M. Worley
Network Administrator • Chris Barrus
Customer Relations • Lee Ann Pham, Susan Pomrantz, Lenell Solomon
Accounting • Paula Garrety, Jan Webber
Warehouse Coordinator • Erik Davidson

Art Direction/Production • InfoGraphix

Board of Advisors • Steven Geller, Blake Park, and Alan Carsrud

PRINTED WITH SOY INK™

This publication is printed on paper with recycled content.

MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 850-P Hampshire Road, Westlake Village, CA 91361-2800. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are $47.00 per year. Canadian subscriptions are $59.00 per year. All other international subscriptions are $97.00 per year. Domestic source code disk subscriptions are $77 per year. All international disk subscriptions are $97.00 a year. Please remit in U.S. funds only. Periodical postage is paid at Thousand Oaks, CA and at additional mailing office.

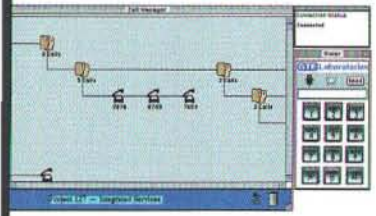
POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.

Contents

February 1998 • Volume 14, No.2

For Macintosh Programmers & Developers

MacTech® MAGAZINE

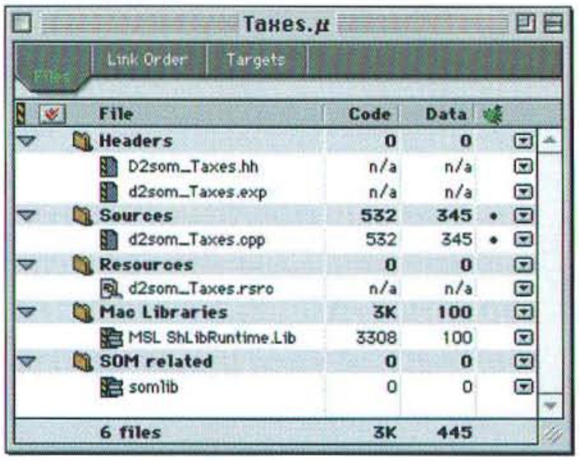


apid Prototyping page 19

Feature Articles

Reader Resources

Columns



Taxes.µ

Files | Link Order | Targets

File	Code	Data
Headers	0	0
D2som_Taxes.hh	n/a	n/a
d2som_Taxes.exp	n/a	n/a
Sources	532	345 •
d2som_Taxes.cpp	532	345 •
Resources	0	0
d2som_Taxes.rsrc	n/a	n/a
Mac Libraries	3K	100
MSL ShLibRuntime.Lib	3308	100
SOM related	0	0
somlib	0	0
6 files	3K	445

And then SOM... page 66

develop Special Section

by Eric Gundrum

Scripting is Part of the Interface

Why do we need scripting languages? Why do we need more than one? I started working in the Mac industry years before we had AppleScript. Back then, a few of the more powerful Mac applications had their own built-in scripting languages, including the communications application I was working on. Then a scripting language was used to automate the internal functions of the application, freeing the user from having to watch over time-consuming tasks, or just assisting with mundane, repetitive work. We did not have anything like the shell script of UNIX or the batch files of DOS to tie applications together.

With the release of System 7, AppleScript entered the scene. Apple had given us a very powerful core language, with all the essential decision and flow control features. Unfortunately, Apple's original release of AppleScript was seriously flawed. The underlying technology was great, but Apple didn't show us how best to use it. They were in such a hurry to get AppleScript out the door, that they hadn't had time to build it into the Finder. Developers often look to the Finder for ideas on how to best make use of Apple technologies, for better or worse. If we want an example of how the menus should look in an application, we look to the Finder. We do the same for ideas on the About box, Balloon Help and many other technologies. Apple did not provide developers with a rich example of how to implement a scripting interface to an application. Developers were slow to adopt AppleScript, and consequently, so were users.

Designing a clean scripting interface to an application is as important as designing a clean graphical interface to the application. Yet few developers are willing to invest even a tenth the time in the scripting interface that they invest in the graphical interface. Most scripting interfaces are implemented at the last minute, usually as the application is going to beta testers. By then there is not sufficient time in the development cycle to change the application to better support scripting. Furthermore, most scripting implementations are designed by programmers, usually with intimate knowledge of the inner workings of the application. They think in terms of the application's internal messages and data structures and figure they can use AppleScript to extend these to the user. Then the documentation folks have to figure out how to explain what the programmers mean by commands like "OpenConnectionToFTPServer(serverAddress, portNumber, showWindow)".

Talk to your Mac

I prefer AppleScript because writing in the AppleScript language is like talking to my computer. Being the asocial geek that I am, I like the idea of my computer understanding me when I talk to it. When I can say things like "Computer, please tell the application "Finder" to open the window named "projects." Or, while in a QuickTime VR world: "Computer, pan left 90 degrees." This sure beats having to say "OpenFinderWindow("projects")". AppleScript provides a framework for communicating with applications this way.

The biggest mistake Apple made with AppleScript was in not giving developers the "Human Scriptability Guidelines," the AppleScript equivalent of the "Human Interface Guidelines." After all, AppleScript is an alternative interface to applications. For that interface to be easy to use, it must be well designed and consistent with other applications. We all know how much users benefit from consistency across applications. Many of us fought hard to convince unenlightened developers who, in the early days of the Mac, neglected to include an Edit menu, used the wrong command key for "Close Window," put the Preferences item in the wrong menu, or otherwise made mistakes in the interface. Making applications scriptable requires this same attention to detail, but the problems are more subtle, and the solutions less obvious. Without Apple's guidance, we are left to figure out for ourselves how best to design a scripting interface, and everyone designs a different look and feel.

In my experience designing the scripting interface before designing the code helps a lot. This way, you can factor your application, building in well defined objects that easily map to your scripting terminology. To help you design your terminology, start by talking to your computer; verbally direct your application to perform common tasks. Ask users familiar with what your application does to write the directions to common tasks as if they were directing a colleague to perform the task. These activities will help you identify the object that users think of when they perform these tasks. Those objects should be the main elements of your terminology. This will also help you think about all the features you must have in your application, and how to organize the underlying code. Then all you have to do is write it. ;-)

In This Issue

Two of the articles in this issue can help you design better scripting interfaces your applications. John Schettino's look at prototyping applications with FaceSpan provides suggestions on designing an application with scripting in mind. Cal Simone and Bill Cheeseman have designed a framework for judging just how scriptable is an application. Their article identifies the essential features of any scriptable application. You also will find articles about scripting environments other than AppleScript. In today's multi-platform world, Perl and tcl/tk are important technologies. In the past, the Macintosh versions of these tools have not kept up with their UNIX counterparts. This is no longer a problem. As you will see from the articles, these Macintosh tools are very powerful. MT

MORE DEVELOPERS PROTECT.

MacHASP Packs More Into Less.

Based on state-of-the-art technology, MacHASP packs the industry's most advanced security, compatibility and flexibility into a compact and end-user-friendly dongle.

Grow With Aladdin!

The fastest growing company in the industry, with over 6 million keys sold to 20 thousand developers worldwide, Aladdin is setting the standard for software security today.

NSTL Study Rates HASP No.1!

A recent test conducted by the National Software Testing Labs[†], the world's foremost independent lab, compared the flagship PC products of leading software protection vendors. The result? HASP was rated the clear overall winner - and number one in all the major comparison categories. For a full copy of the NSTL report, contact your local HASP distributor.

MacHASP® PROTECTS MORE.

Mac™OS

These days, more and more developers are choosing to protect their software against piracy. They're protecting more products, on more platforms, with better protection – and selling more as a result.

And more of these developers are protecting with MacHASP. Why? Because MacHASP offers more security, more reliability and more features than any other product on the market. Only MacHASP offers capabilities such as network support, anti-debugging envelope protection, and secure remote activation and updating.

MacHASP supports all 68030, 68040 and PowerPC-based Macs, and compatibles and Mac operating systems from version 6.5 to System 8. As a licensed Apple Partner, Aladdin will continue to support future OS developments with guaranteed full and transparent compatibility to the ADB standard. AppleTalk and OpenTransport networks are fully supported with comprehensive monitoring facilities.

To learn more about how you can protect better – and sell more – call now to order your MacHASP Developer's Kit.

The MacHASP Developer's Kit contains everything you need to protect your software today!

1-800-223-4277

www.aks.com

ALADDIN™

The Professional's Choice

North America	**Aladdin Knowledge Systems Inc.** Tel: 800 223-4277, 212 564-5678, Fax: 212 564 3377, Email: hasp.sales@us.aks.com
Int'l Office	**Aladdin Knowledge Systems Ltd.** Tel: +972 3 636-2222, Fax: +972 3 537-5796, Email: hasp.sales@aks.com
Germany	**Aladdin Germany** Tel: +49 89 89-42-21-37, Fax: +49 89 89-42-21-40, Email: info@fast.ag.de
UK	**Aladdin Knowledge Systems UK Ltd.** Tel: +44 1753 622-266, Fax: +44 1753 622-262, Email: sales@aldn.co.uk
Japan	**Aladdin Japan Co., Ltd.** Tel: +81 426 60-7191, Fax: +81 426 60-7194, Email: sales@aladdin.co.jp
Benelux	**Aladdin Software Security Benelux B.V.** Tel: +31 24 648-8444, Fax: +31 24 645-1981, Email: aladdin@worldaccess.nl
France	**Aladdin France SA** Tel: +33 1 41-37-70-30, Fax: +33 1 41-37-70-39, Email: info@aladdin.fr
Russia	**Aladdin Software Security R.D. Ltd.** +7 095 923-0588, Fax: +7 095 928-6781, Email: hasp@aladdin.ru

■ **Australia** Conlab 03 98985685 ■ **China** Feitian 010 62567389 ■ **Czech** Atlas 02 766085 ■ **Denmark** Berendsen 039 577316 ■ **Egypt** Zeineldein 02 3604632 ■ **Finland** ID-Systems 09 8703520 ■ **Greece** Unibrain 01 6756320 ■ **Hong Kong** Hastings 02 5484629
■ **India** Solution 011 2148254 ■ **Italy** Partner Data 02 26147380 ■ **Korea** Dae-A 02 8484481 ■ **Mexico** SiSoft 091 80055203 ■ **Poland** Systherm 061 480273 ■ **Portugal** Futurmatica 01 4116269 ■ **Romania** Ro Interactive 064 140383 ■ **Singapore** ITR 065 5666788
■ **South Africa** D Le Roux 011 8864704 ■ **Spain** PC Hardware 03 4493193 ■ **Sweden** Kordab 455 307 300 ■ **Switzerland** Opag 061 7169222 ■ **Taiwan** Teco 02 5559676 ■ **Turkey** Mikrobeta 0312 4670635 ■ **Yugoslavia** Asys 021 623928

by Dave Mark, ©1997, All Rights Reserved.

Window-Related Events

How a Mac program handles windows

In last month's column, we covered the basics of event loop programming, focusing on four specific events: the mouseDown, mouseUp, keyDown, and autoKey events. If you haven't already, be sure to read through the Event Manager chapter in Inside Macintosh: Macintosh Toolbox Essentials. Pay specific attention to the sections that describe the mouseDown, mouseUp, keyDown, and autoKey events in detail. Finally, go back to last month's program and flesh it out a little. When a mouseDown occurs, draw a string in the event window that describes where and when the mouseDown occurred. For a keyDown, draw the character and key codes embedded in the EventRecord.

In this month's column, we're going to expand our event handling repertoire, focusing on activate, update, and suspend/resume events.

Activate Events

Every time your application creates a window, that window is added to a list maintained by the Window Manager. The windows in this list are in the order that they appear on the screen, from the frontmost to the rearmost.

The frontmost window is also known as the active window. In **Figure 1**, Window #1 starts off as the active window. Notice the difference between the title bars of the active and non-active window.

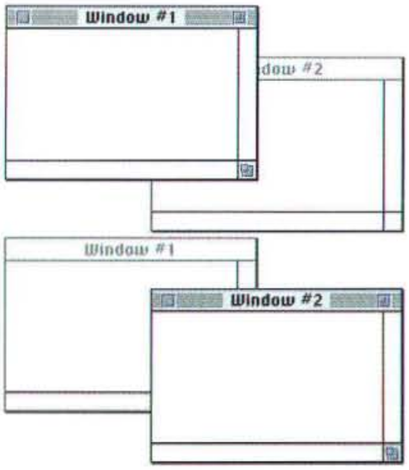


Figure 1. *A mouse click in Window #2 brings it to the front.*

When the mouse is clicked in the rear window, Window #1 is made inactive, then the rear window, Window #2, is made active. Since the front window is made inactive before another window is brought to the front, there will never be more than one active window at a time!

The Window Manager accomplishes this by sending your application a series of activate events. If a rear window is being brought to the front, your application will receive two activate events. The first tells you that the frontmost window is becoming inactive. This event is also known as a deactivate event. The second event tells your application that a window is becoming active. Both of these events set EventRecord.what to activateEvt.

If a window is created, and there are no currently open windows, the Window Manager only generates a single activate event, indicating that the newly created window is becoming active. In this case, no deactivate event is sent.

In general, you'll use activate events if you treat window contents differently for an active window than for an inactive window. For example, your application might highlight selected text in the active window, but not in an inactive window. Activate events are provided for your benefit. Use them as you see fit. You'll see how we discriminate between an activate and deactivate event when we get to our program later in the column.

Update Events

Next on our list of events is an event that tells your program to update the contents of a window. The updateEvt is generated when a window is created (after the activateEvt is generated) and when a new portion of a window is revealed. **Figure 2** shows a typical sequence that generates an updateEvt. The first picture shows Window #1 partially obscuring Window #2. Next, Window #1 is dragged to the left, revealing a previously hidden section of the Window #2. The Window Manager sends an updateEvt to your application, telling it to update the contents of Window #2. The third picture shows the window, after the program responded to the update event.

Aladdin Systems

- Custom Destinations
- Installation from FTP sites
- Install or Uninstall
- Best Compression Available
- Resource Installation
- Built-In Resource Compression
- Full Scripting and Recording
- Shrinkwrap
- Built-In Updaters
- Electronic Transaction Processing

InstallerMaker puts all the pieces together... even from an FTP site!

There are more and more reasons StuffIt InstallerMaker 4.6 is the Complete Installation Solution. Now your installer can automatically retrieve files from an FTP site, so your users always get the right software.

StuffIt InstallerMaker 4.6 picks up where your compiler stops. It allows you to easily create custom installers for distributing software or files. Using our installer reduces technical support calls due to end-user errors during installation, saves disks (if distributing on floppies) or download time (if distributing on-line).

Download a free, fully-functional copy of StuffIt InstallerMaker 4.6 from **www.aladdinsys.com**, or call (408) 761-6200 and ask for Developer Sales.

StuffIt INSTALLERMAKER 4.6

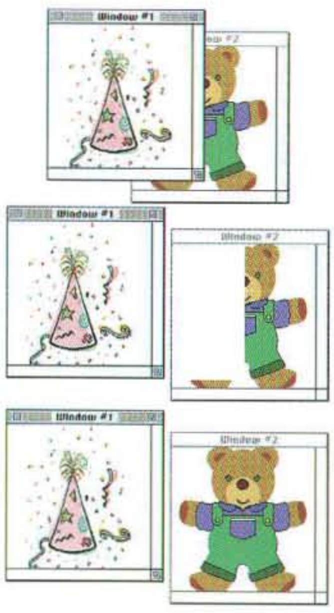


Figure 2. When Window #1 is dragged to the left, more of Window #2 is exposed, causing an update event.

Figure 3 shows a slightly different sequence, involving update and activate events. This time, the mouse is clicked in Window #2, bringing it to the front. First, a deactivate event is generated for Window #1. Next, an activate event is generated for Window #2. Finally, an update event is generated, asking the program to update the contents of Window #2.

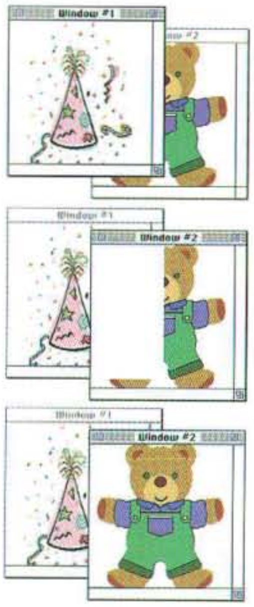


Figure 3. When Window #2 is moved to the front, two activate events and an update event are generated.

Once again, you'll see update and activate events in action in the program later in the column.

Suspend & Resume Events

When you click in a window belonging to a background application, the active application receives a suspend event, and the background application receives a resume event and is moved to the foreground.

You might use suspend and resume events to determine the actions taken by your program. For example, in the foreground, you might display a special tool palette, or perhaps run an animation. When your application moves into the background, you'll receive a suspend event and you might hide the tool palette, or discontinue the animation until you get a resume event.

It's important to note that your application can continue running, even if it receives a suspend event. It will still continue to receive update (and other appropriate events) while in the background. At the very least, it's a good idea to treat a suspend event as you would a deactivate event when it comes to the contents of your windows. For example, if you usually deselect any selected text when a window is deactivated, do the same thing for the active window when you receive a suspend event.

At Last! The Program!

As promised, here's a program that incorporates all the events described in this column. WindowMaster creates a single window and handles all the usual events relating to windows. **Figure 4** shows the WindowMaster window in all its glory. Notice that this window sports a close box, a zoom box, a drag region (the title bar), and a grow box. Though this window doesn't support scroll bars, there is room for them. In most cases, if a window has a grow box, it has room for scroll bars.

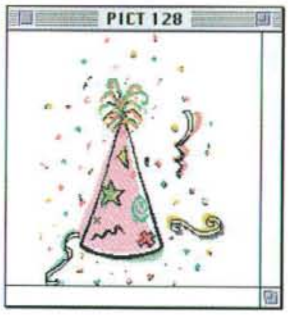


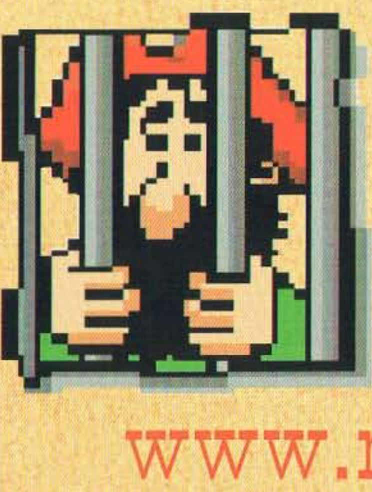
Figure 4. The WindowMaster window

Creating the WindowMaster Resources

Create a folder in your development folder named WindowMaster. Launch ResEdit, then create a resource file named WindowMaster.rsrc in your WindowMaster folder. You'll create two resources in this file.

First, create a WIND resource according to the specifications shown in **Figure 5**. Next, select Set 'WIND' Characteristics... from the WIND menu and set the window's title to PICT 128. Next, copy a graphic from your scrapbook into the clipboard. If you don't have anything interesting in your scrapbook, go draw something. I'll wait.

REWARD
$15,000,000,000



Have you seen this man ?

Hacker Harry and the pirate gang are loose and robbing the software stagecoach. Whether it's software, bank account information, money transactions, or customer databases, if you want to stop them and collect your part of the reward... the place to find your New Generation Protection solution is at:

www.micromacro.com

The expected losses due to piracy in the software industry alone are over $15 billion for 1997. Some of this belongs to YOU. By using MicroGuard's New Generation Copy Protection keys, you will enforce your rights and guarantee your profits.

International
Micro Macro Technologies, Ltd.
3 Hashikma St.
P.O.Box 11516, Azur 58001
Israel

Tel: (972-3) 558-2345
Fax: (972-3) 558-2344
E-mail: info@micromacro.com

USA
MicroGuard
631 South Pontiac St.
Denver, CO 80224
USA

Tel: (303) 320-1628
Fax: (303) 320-1599
E-mail: usa@micromacro.com

SEE YOU AT:
COMDEX Fall '97
November 17-21 1997
LAS VEGAS
Booth #S8049

MICROSOFT WINDOWS COMPATIBLE

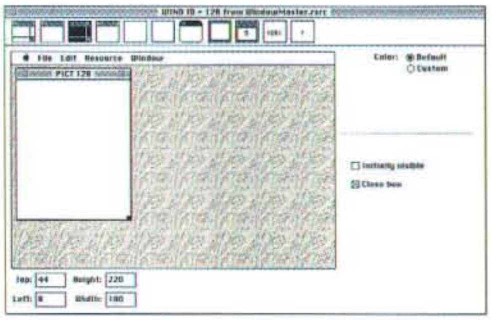
Mac OS

Charter member Pirate Busters

MICROGUARD™

If you are running Mac OS 8, you can give your window the Mac OS 8 appearance with a few extra steps. First we have to configure ResEdit to support a Mac OS 8 window. See the row of window icons across the top of **Figure 5**? These icons select the window's procID, which tells the system how to select and configure the WDEF that makes the window. Double click one of the icons containing a questionmark shown at the end of the row. In the resulting window, enter 1031 in one of the empty fields. Now one of your icons will contain 1031 as in **Figure 5**. Select this icon. If you are doing this under Mac OS 8, you will see a Mac OS 8-style window. You can experiment with other document window procIDs ranging from 1024 to 1031. These are all listed in Appearance.h of the Appearance Manager SDK.

Once you've got a graphic in the clipboard, return to ResEdit and select Paste from the Edit menu. ResEdit will place your picture in a PICT resource. Make sure the PICT resource has a resource ID of 128. Well, that's it for ResEdit. Quit, making sure you save your changes.


WIND ID = 128 from WindowMaster.rsrc
File Edit Resource Window
PICT 128

Color: [✓ Default] Custom

☐ Initially visible
☑ Close box

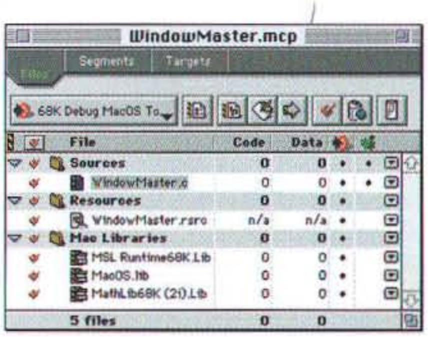
Top: 44 Height: 220
Left: 8 Width: 180

Figure 5. *Specifications for WindowMaster's WIND resource.*

Creating the WindowMaster Project

Once you're out of ResEdit, launch CodeWarrior and create a new project based on the MacOS:C/C++:Basic Toolbox 68k stationary. Turn off the Create Folder check box. Name the project WindowMaster.mcp and place it in your WindowMaster folder. Remove SillyBalls.c and SillyBalls.rsrc from the project; we will not be using these files. From the Finder, drag and drop your WindowMaster.rsrc file into the project window. You also can remove the ANSI Libraries group from the project, because we won't need them, either.

Select New from the File menu to create a new window. Save it under the name WindowMaster.c, Select Add Window from the Project menu to add WindowMaster.c to the project. Your project window should look something like **Figure 6**.



File	Code	Data
Sources	0	0
WindowMaster.c	0	0
Resources	0	0
WindowMaster.rsrc	n/a	n/a
Mac Libraries	0	0
MSL Runtime68K.Lib	0	0
MacOS.lib	0	0
MathLib68K (2i).Lib	0	0
5 files	0	0

Figure 6. *WindowMaster project window.*

Rather than print the code here twice, we'll go straight to the walk-through. You can type in the code as we discuss it below and you will end up with the complete program, or you can save your fingures some effort and get the complete project from MacTech's ftp site <ftp://ftp.mactech.com/src/>.

Walking Through the Source Code

Just like last month's program, WindowMaster is based on an event loop architecture. Also, like last month's column, the program starts off by including <Sounds.h> to access the SysBeep() system call. This month we add <limits.h> to define SHRT_MAX.

```
#include <limits.h>
#include <Sound.h>

#define kBaseResID        128
#define kMoveToFront      (WindowPtr)-1L
#define kSleep            7

#define kScrollBarAdjust (16-1)
#define kLeaveWhereItIs  false
#define kNormalUpdates   true
#define kMinWindowHeight 50

#define kMinWindowWidth  80
```

gDone is initialized to false, then set to true when a click occurs in the window's close box. Note that this is not the way Mac applications normally exit, but we haven't got to menus yet, so a click in the close box will have to do for now.

```
/******************************** Globals *********/

Boolean    gDone;
```

As always, the code includes prototypes for all functions.

```
/******************************** Function Prototypes    ****/

void ToolBoxInit( void );
void WindowInit( void );
void EventLoop( void );
void DoEvent( EventRecord *eventPtr );
void HandleMouseDown( EventRecord *eventPtr );
void DoUpdate( EventRecord *eventPtr );
void DoPicture( WindowPtr window, PicHandle picture );
void DoActivate( WindowPtr window, Boolean becomingActive );
void DoSuspendResume( Boolean resuming );
void CenterPict( PicHandle picture, Rect *srcRectPtr,
          Rect *destRectPtr );
```

main() initializes the Toolbox, creates a window, then enters the main event loop.

```
/******************************** main *********/

void main( void )
{
  ToolBoxInit();
  WindowInit();

  EventLoop();
}
```

ToolBoxInit() is the same as it ever was.

```
/************************************ ToolBoxInit */

void ToolBoxInit( void )
{
  InitGraf( &qd.thePort );
```

"Without a doubt, the Premiere Resource Editor for the Mac OS ... A wealth of time-saving tools."
– MacUser Magazine Eddy Awards

"A distinct improvement over Apple's ResEdit."
– MacTech Magazine

"Every Mac OS developer should own a copy of Resorcerer."
– Leonard Rosenthol, Aladdin Systems

"Without Resorcerer, our localization efforts would look like a Tower of Babel. Don't do product without it!"
– Greg Galanos, CEO and President, Metrowerks

"Resorcerer's data template system is amazing."
– Bill Goodman, author of Smaller Installer and Compact Pro

"Resorcerer Rocks! Buy it, you will NOT regret it."
– Joe Zobkiw, author of ***A Fragment of Your Imagination***

"Resorcerer will pay for itself many times over in saved time and effort."
– MacUser review

"The template that disassembles 'PICT's is awesome!"
– Bill Steinberg, author of Pyro! and PBTools

"Resorcerer proved indispensible in its own creation!"
– Doug McKenna, author of Resorcerer

MACUSER EDITORS' CHOICE AWARD 1993 WINNER

Accelerated for Power Macintosh

Resorcerer® 2

Version 2.0

The Resource Editor for the Mac™ OS Wizard

ORDERING INFO

Requires System 7.0 or greater, 1.5MB RAM, CD-ROM

Standard price: $256 (decimal)
Website price: $128 - $256
(Educational, quantity, or other discounts available)

Includes: Electronic documentation
60-day Money-Back Guarantee
Domestic standard shipping

Payment: Check, PO's, or Visa/MC
Taxes: Colorado customers only

Extras (call, fax, or email us):
COD, FedEx, UPS Blue/Red,
International Shipping

MATHEMÆSTHETICS, INC.
PO Box 298
Boulder, CO 80306-0298 USA
Phone: (303) 440-0707
Fax: (303) 440-0504
resorcerer@mathemaesthetics.com

New in 2.0:

- **Very fast, HFS browser for viewing file tree of all volumes**
- **Extensibility for new Resorcerer Apprentices (CFM plug-ins)**
- **New AppleScript Dictionary ('aete') Apprentice Editor**
- **MacOS 8 Appearance Manager-savvy Control Editor**
- **PowerPlant text traits and menu command support**
- **Complete AIFF sound file disassembly template**
- **Big-, little-, and even mixed-endian template parsing**
- **Auto-backup during file saves; folder attribute editing**
- **Ships with PowerPC native, fat, and 68K versions**

- Fully supported; it's easier, faster, and more productive than ResEdit
- Safer memory-based, not disk-file-based, design and operation
- All file information and common commands in one easy-to-use window
- Compares resource files, and even **edits your data forks** as well
- Visible, accumulating, editable scrap
- Searches and opens/marks/selects resources by text content
- Makes global resource ID or type changes easily and safely
- Builds resource files from simple Rez-like scripts
- Most editors DeRez directly to the clipboard
- All graphic editors support screen-copying or partial screen-copying
- Hot-linking Value Converter for editing 32 bits in a dozen formats
- Its own 32-bit List Mgr can open and edit very large data structures
- Templates can pre- and post-process any arbitrary data structure
- Includes nearly 200 templates for common system resources
- TMPLs for Installer, MacApp, QT, Balloons, AppleEvent, GX, etc.
- Full integrated support for editing color dialogs and menus
- Try out balloons, 'ictb's, lists and popups, even create C source code
- Integrated single-window Hex/Code Editor, with patching, searching
- Editors for cursors, versions, pictures, bundles, and lots more
- Relied on by thousands of Macintosh developers around the world

To order by credit card, or to get the latest news, bug fixes, updates, and apprentices, visit our website...

www.mathemaesthetics.com

```
  InitFonts();
  InitWindows();
  InitMenus();
  TEInit();
  InitDialogs( nil );
  InitCursor();
}
```

WindowInit() calls GetNewWindow() to load the WIND resource from the resource file.

```
/****************************** WindowInit ********/

void WindowInit( void )
{
  WindowPtr  window;

  window = GetNewWindow( kBaseResID, nil, kMoveToFront );
```

If the resource wasn't found, beep once, then exit.

```
  if ( window == nil )
  {
    SysBeep( 10 );  /* Couldn't load the WIND resource!!!*/
    ExitToShell();
  }
```

Once the window is created, make it visible, then make it the current port. Notice that this routine does not do any drawing. We'll draw our picture in response to an update event.

```
  ShowWindow( window );
  SetPort( window );
}
```

EventLoop() sets gDone to false, then loops around a call to WaitNextEvent(). If WaitNextEvent() returns true, the event is passed to DoEvent().

```
/****************************** EventLoop ********/

void EventLoop( void )
{
  EventRecord  event;

  gDone = false;
  while ( gDone == false )
  {
    if ( WaitNextEvent( everyEvent, &event, kSleep, nil ) )
      DoEvent( &event );
  }
}
```

DoEvent() switches on the event's what field.

```
/*********************************** DoEvent  */

void DoEvent( EventRecord *eventPtr )
{
  Boolean  becomingActive, resuming;
  switch ( eventPtr->what )
  {
```

A mouseDown event is passed to HandleMouseDown(). An updateEvt is passed on to DoUpdate().

```
    case mouseDown:
      HandleMouseDown( eventPtr );
      break;
    case updateEvt:
      DoUpdate( eventPtr );
      break;
```

In the case of an activate event, the event's modifiers field holds the key to whether the event is an activate or deactivate event. activeFlag is a mask that designates one of the bits in the modifiers field. If the bit is set, the event is an activate event. If the bit is clear, the event is a deactivate event. The Boolean becomingActive is true if the event is an activate event. Once becomingActive is set, it is passed on to DoActivate(). The event's message field holds a pointer to the window being activated or deactivated.

```
    case activateEvt:
      becomingActive = ( (eventPtr->modifiers & activeFlag)
                        == activeFlag );
      DoActivate( (WindowPtr)eventPtr->message,
          becomingActive );
      break;
```

Similarly, an osEvt is used to indicate either a suspend or resume event. suspendResumeMessage is a predefined constant that designates the suspend/resume bit in the message field. resuming is set to true if the event is a resume event. Once set, resuming is passed on to DoSuspendResume().

```
    case osEvt:
      resuming = ( eventPtr->message & suspendResumeMessage )
                    == resumeFlag;
      DoSuspendResume( resuming );
      break;
  }
}
```

HandleMouseDown() handles the mouseDown event.

```
/*********************************** HandleMouseDown */

void HandleMouseDown( EventRecord *eventPtr )
{
  WindowPtr  window;
  short      thePart;
  GrafPtr    oldPort;
  long       windSize;
  Rect       growRect;
```

FindWindow() takes the event's where field and returns the window at those coordinates. thePart indicates the part of the window the mouse click occurred in.

```
  thePart = FindWindow( eventPtr->where, &window );
```

If the click was in a portion of the screen not belonging to our application (like a desk accessory window), thePart is set to inSysWindow and we'll pass the event back to the system with SystemClick().

```
  switch ( thePart )
  {
    case inSysWindow :
      SystemClick( eventPtr, window );
      break;
```

If the event was in the content region of the window, we'll typically call SelectWindow() to bring the window to the front. Since we only have one window, this line isn't particularly useful. Later on though, you'll add another window to this program and you'll want to keep this code in here.

As your windows get more complex, you'll want to do more with inContent clicks than just select the window. You might have a button or scroll bar in the window that needs action, or you might have some text that needs selection. This is the jumping off point for all clicks that occur in the window. Eventually, we'll add a routine named DoContentClick() to process clicks in a window's content region.

```
case inContent:
  SelectWindow( window );
  break;
```

If the click was in the window's drag region (title bar), we'll pass the window (retrieved by FindWindow()), and the mouse click coordinates to the Toolbox routine DragWindow(). The third parameter is a bounding rectangle that determines where on the screen the window may be dragged. screenBits.bounds is a System global variable that defines the boundaries of the main display.

If you have two monitors, this code works fine, however. DragWindow() checks for screenBits.bounds as a parameter and, if it finds it, allows dragging anywhere on any monitor attached to the system.

```
case inDrag :
  DragWindow( window, eventPtr->where,
      &qd.screenBits.bounds );
  break;
```

If the mouseDown was in the close box, we'll call TrackGoAway() to see if the mouse was released while still inside the close box. If so, gDone is set to true. TrackGoAway() is the routine that does the little animation in the close box.

```
case inGoAway :
  if ( TrackGoAway( window, eventPtr->where ) )
    gDone = true;
  break;
```

If the click was in the grow box, we set up a rectangle that defines how large and how small the window is allowed to grow. Basically, it's good policy to put a limit on how small a window can get, but, unless you've got a pressing reason, you should allow windows to get as large as the user wants.

```
case inGrow:
  growRect.top = kMinWindowHeight;
  growRect.left = kMinWindowWidth;
  growRect.bottom = SHRT_MAX;
  growRect.right = SHRT_MAX;
```

Next, this rectangle is passed on to GrowWindow(), which tracks the mouse, allowing the user to specify the new window size.

```
  windSize = GrowWindow( window, eventPtr->where,
      &growRect );
```

The return value contains two 2-byte values, indicating the new height and width of the window. If both are zero, the user has not changed the window size.

```
  if ( windSize != 0 )
  {
```

First, we'll save the current port (in case it's not this window), then make this window the current port. Next, we erase the entire window and change the window's size by calling SizeWindow(). The last parameter tells the system we want this resizing to generate an update event.

RHAPSODY READY POWER

NOW

THERE'S NO REASON TO WAIT

POWER YOUR RHAPSODY APPLICATIONS WITH OPENBASE – the high-performance SQL database server for Rhapsody! Tap into proven technology, it's ready now!

RELEASE THE POWER – make applications that are scalable, multi-user and fast with OpenBase.

- Aggressive Multi-threading
- Text Searching
- Change Notification
- Row Level Locking
- Variable Record Length Technology

THE POWER TO BUILD – make building applications easy with the full line of graphical tools, native EOF adaptors, and Rhapsody source code examples that come with OpenBase.

THE POWER IS YOURS

Order now and take advantage of our

FREE SINGLE-USER RUNTIMES

SPECIAL OFFER 5 CONNECTIONS JUST $495!

OPENBASE® INTERNATIONAL

http://www.openbase.com

603.547.8404 / info@openbase.com

```
        GetPort( &oldPort );
        SetPort( window );
        EraseRect( &window->portRect );
        SizeWindow( window, LoWord( windSize ),
            HiWord( windSize ), kNormalUpdates );
```

Next, we call **InvalRect()** to tell the system that the entire window should be redrawn with the next update event, not just the new area revealed by the grow box. In fact, this call will force an update even if the window was shrunk.

```
        InvalRect( &window->portRect );
```

Because we center our picture in the window, we need to redraw the window contents whenever the window changes size. To see why this is true, try commenting out the previous line of code.

Update events are tricky. It is definitely worth reading the section of *Inside Macintosh: Macintosh Toolbox Essentials* that covers the Window Manager. You might also check out Chapter 4's Updater program in Volume I of the *Mac Primer.*

Once the window is resized, the port is set back to its old value. Since **InvalRect()** applies to the current port, it was important that we make the window being grown the current port.

```
        SetPort( oldPort );
      }
      break;
```

Finally, a click in the zoom box follows a similar strategy. Again, **TrackBox()** is called to verify that the mouse was released inside the zoom box. If so, **ZoomWindow()** is called to zoom the window, or return it to its old position, depending on its current state.

```
    case inZoomIn:
    case inZoomOut:
      if ( TrackBox( window, eventPtr->where, thePart ) )
      {
        GetPort( &oldPort );
        SetPort( window );
        EraseRect( &window->portRect );
        ZoomWindow( window, thePart, kLeaveWhereItIs );
        InvalRect( &window->portRect );
        SetPort( oldPort );
      }
      break;
  }
}
```

DoUpdate() is called whenever an update event occurs. First, the window is retrieved from the event's **message** field. Next, **BeginUpdate()** is called, telling the Window Manager that we're handling updates for this window. **BeginUpdate()** will restrict drawing to the region of the window that needs updating, also known as the update region. If you call **InvalRect()** on the whole window, the whole window is available for drawing.

```
/************************************ DoUpdate    */

void DoUpdate( EventRecord *eventPtr )
{
  PicHandle  picture;

  WindowPtr  window;
  window = (WindowPtr)eventPtr->message;

  BeginUpdate( window );
```

Next, we load the picture and pass it on to DoPicture().

```
  picture = GetPicture( kBaseResID );
  if ( picture == nil )
  {
    SysBeep( 10 );  /* Couldn't load the PICT resource!!! */
    ExitToShell();
  }
  DoPicture( window, picture );
  EndUpdate( window );
}
```

DoPicture() makes the window the current port, then erases the entire window. Note that this will already have been done if the update event was caused by a resize or zoom, but there's no harm in this minor duplication of code.

```
/****************************** DoPicture *********/

void DoPicture( WindowPtr window, PicHandle picture )
{
  Rect       drawingClipRect, destRect;
  RgnHandle  tempRgn;
  SetPort( window );
  EraseRect( &window->portRect );
```

Next, a new region is created. We pass this region handle on to **GetClip()**, which places a copy of the window's clipping region in **tempRgn**. The clipping region defines which parts of the window can be drawn in and which parts can't. Again, it is a good idea to read the entire Window Manager chapter in *Inside Macintosh: Macintosh Toolbox Essentials*, as there's not enough space to cover all of it in this column.

```
  tempRgn = NewRgn();
  GetClip( tempRgn );
```

Since we want to center the picture in the part of the window not covered by the grow region and scroll bar areas, we'll subtract those areas from the window's portRect. Next, we'll pass this rectangle on to **ClipRect()** making it the new clip region.

```
  drawingClipRect = window->portRect;
  drawingClipRect.right -= kScrollBarAdjust;
  drawingClipRect.bottom -= kScrollBarAdjust;
  ClipRect( &drawingClipRect );
```

Next, we'll call **CenterPict()** to center the picture in **drawingClipRect**, returning the rectangle the picture should be drawn in as its third parameter. We then draw the picture with **DrawPicture()**.

```
  CenterPict( picture, &drawingClipRect, &destRect );
  DrawPicture( picture, &destRect );
```

Next, the clipping region is set back to its old value and the grow icon is redrawn, completing our update.

```
  SetClip( tempRgn );
  DisposeRgn( tempRgn );

  DrawGrowIcon( window );
}
```

It's important to remember that your application can receive update events even if it's running in the background. If a foreground application's window covers your application's windows and is then moved, the Window Manager will send your application update events asking you to redraw the affected windows.

DoActivate() is pretty simple. It redraws the grow icon in the specified window. DrawGrowIcon() will draw a different grow icon, depending on whether your window is frontmost or not. When your application starts handling more than one window, you'll start making use of the second parameter, becomingActive. For now, we use the CodeWarrior #pragma unused directive to let the compiler know we intentionally are not using this variable. If we didn't add this line, CodeWarrior would warn us that the variable was never used.

```
/************************************ DoActivate   */

void DoActivate( WindowPtr window, Boolean becomingActive )
{
  #pragma unused ( becomingActive )
  DrawGrowIcon( window );
}
```

DoSuspendResume() is also pretty simple. This time the frontmost window's grow icon is redrawn. A front window, which has been suspended gets the same grow icon as any other window. DrawGrowIcon() is smart enough to know the state of your application.

Once again, as your programs get more complex, you'll make use of the second parameter, resuming.

```
/************************************ DoSuspendResume */

void DoSuspendResume( Boolean resuming )
{
  #pragma unused ( resuming )
  WindowPtr  window;
  window = FrontWindow();
  DrawGrowIcon( window );
}
```

CenterPict() centers the picture in the rectangle pointed to by srcRectPtr, returning the result in the rectangle pointed to by destRectPtr.

```
/****************** CenterPict ********************/

void CenterPict( PicHandle picture, Rect *srcRectPtr,
                 Rect *destRectPtr )
{
  Rect pictRect;

  pictRect = (**( picture )).picFrame;

  OffsetRect( &pictRect, srcRectPtr->left - pictRect.left,
                 srcRectPtr->top   - pictRect.top);
  OffsetRect( &pictRect,
       (srcRectPtr->right - pictRect.right)/2,
       (srcRectPtr->bottom - pictRect.bottom)/2);

  *destRectPtr = pictRect;
}
```

RUNNING WINDOWMASTER

Select Run from the Project menu to run WindowMaster. A single window, like the one shown in Figure 4 should appear. Click the mouse in the title bar and drag the window around on the screen. Next, click in the grow box and resize the window. Try making it tall and skinny, then short and fat. Notice that there is a limit to how small you can make the window. Next, click the zoom box in the upper-right corner of the window. The window should expand to fill the main screen. Click the zoom-box again to return the window to its previous size.

IT'S NOT JUST FOR BROWSERS

Add Macro Capabilities to your Applications with JavaScript/ECMAScript

***ScriptEase*:Integration SDK**

gives you

- More flexible applications
- Faster application development
- Easier portability
- Improved code reuse

Nombas Inc.
64 Salem Street, Medford, MA 02155
Tel:(888)766-6227 Fax:(781)391-3842
Email: nombas@nombas.com

www.nombas.com

(JavaScript is a trademark of Sun Microsystems, Inc.)

Next, drag the window downwards, so it is halfway off the screen, obscuring part of the picture. Now drag it back up. The picture will be redrawn in full. Finally, click in the window's close box to exit the program.

TILL NEXT MONTH

There's a lot going on in this program. As I've said before, do yourself a favor and read the Window Manager chapter in *Inside Macintosh: Macintosh Toolbox Essentials*. If a concept in this program seems fuzzy, try commenting out some of the code to see what happens. This is especially useful with the code that affects or handles update events. You might also try commenting out the calls to DrawGrowIcon() just to see what happens.

Finally, try adding another window to the program. Add a second WIND and a second PICT resource to the resource file, then create the new window in WindowInit(). You'll be amazed how easy it is to add a second window. Go ahead, try it! MT

Visit MacTech® Magazine's Web site!
http://www.mactech.com

by Cal Simone and Bill Cheeseman

The AppleScript Scorecard Guidelines

Is your application truly scriptable? These guidelines may help you find the answer

For users writing scripts in AppleScript, the number one challenge is the lack of intuitiveness in scripting implementations and consistency when moving among scriptable applications of different types. To aid potential users of scriptable applications in determining the suitability of particular products for their scripting scenarios and, more importantly, to promote these characteristics among developers of scriptable applications, a set of guidelines has been developed, in the form of an AppleScript "Scorecard".

By the time you read this, the Catalog of Scriptable Products should be available on the Web, tentatively at <http://www.mainevent.com/scorecard/>. The Catalog is a comprehensive directory of scriptable applications, and includes reviews and ratings. The guidelines put forth in this article were developed to form the basis for the reviews and ratings found in the Catalog.

Although the guidelines were developed primarily as an aid for users, they are offered here so that, as the developer of a scriptable product, you can see, in broad terms, what makes up an application that delivers the power of user scripting into the hands of users. Note that, at the time of writing this, we were close to completion, but the final point distribution had not been frozen.

Even if you don't submit your application for a rating, following the criteria of this scoring system will guide you to making your application more easily scripted and more consistent with other well-designed object model-based applications.

THE GUIDELINES

To promote full AppleScript support in the widest range of products, an application is rated in the AppleScript Scorecard in part for the full and proper implementation of specific scripting features, such as the object model and recordability. To promote the highest quality, an application is also rated on a more qualitative assessment of the design, completeness, transparency, stability and documentation of its AppleScript implementation. In the first phase of rating an application, a range of points is awarded for the presence of specific features, taking into account the extent to which each feature is implemented and its conformity to Apple standards and conventions. In the second phase, additional points are awarded on a sliding scale for specific overall qualities of the scripting implementation.

Cal Simone <cal@script.org>, champion of AppleScript, is the President of Main Event Software <mainevent@his.com> and the designer of the Scripter authoring and development environment for AppleScript. In the first half of 1997, Cal spearheaded a 5-month effort to rescue AppleScript from the chopping block during Apple's "technology evaluation" (also known as the "spring cleaning") and to convince Apple's top management to commit to AppleScript in Rhapsody.

Bill Cheeseman <cheeseb@mediaone.net> is a lawyer in Massachusetts, and he maintains The AppleScript SourceBook site at <http://oasis.bellevue.k12.wa.us/cheeseb/index.html>. A true-life legal experience of his is the subject of the best-selling book "A Civil Action," and it is being made into a Hollywood movie, starring John Travolta, to be released in late 1998. Bill has been programming Apple computers as a hobby since 1977.

The AppleScript Scorecard rates an application's support for AppleScript on a scale of 0 to 100. The final score is converted to a scale of one to five "scripts", with half-scripts (1-10 = 1/2 script, 11-20 = 1 script, etc.). The "scripts" score is the primary rating for public distribution, but the points score can be published in parentheses (after converting it to a scale of 0.0 to 10.0 with one decimal place). An example: "3-1/2 scripts (7.2)".

An application is not considered scriptable, and it is not rated, if it has no dictionary or terminology ('aete') resource, or if it has a dictionary that supports only the Required Suite (the run, open, print and quit Apple events) and the "do script" event. Its Finder "scriptable" property may be true because it does respond to the Required Suite, but this by itself does not reach the threshold necessary to be rated.

To be considered for rating, therefore, an application must implement a meaningful and reasonably useful set of verbs in addition to those in the Required Suite (run, open, print and quit), although not necessarily in the form of additional, named suites. Even a small number of supported commands suffices, as long as they enable a scripter to take advantage of some significant and useful functionality of the application. The "do script" event, without more, does not qualify (because this is a rating of the application's AppleScript features, not of a proprietary internal scripting system.)

Phase 1 — Scripting Features Supported

Points are awarded in Phase 1 for the presence of the specific AppleScript features described below. A range of points may be awarded for the presence of each feature to take into account the extent of its implementation and its conformity to Apple standards and conventions, but Phase 1 does not otherwise rate qualitative aspects of the application's AppleScript support.

Implementation of any of these AppleScript features in conformity to its description here merits awarding the maximum number of points for that feature. Points should be deducted from the maximum for failure to implement specific aspects of a feature where indicated. Points should also be subtracted for limiting a scripting feature such as the object model to a narrow subset of an application's functionality; the penalty should be roughly proportional to the fraction of the application's functionality that is not covered by the scripting implementation. The maximum score for complete implementation of all possible scripting features is 40 points.

Object Model Support — supports the Apple event object model — 0 to 25 points

The application supports the object model. It includes at least the essential verbs in the Core, or Standard, Suite ("make", "get" and "set," at a minimum). It also implements a meaningful set of objects that can contain other objects and that have properties. The elements contained in objects have a variety of reference forms (e.g., by index, by name, by unique ID) suitable to their intended use. Its supported commands, including new verbs devised for the application, can operate on various objects and properties in object model fashion. For example, instead of providing commands that are verb-noun hybrids like "GetColor", a graphics application might support objects of several classes, including "picture," which can have any of several properties, including "color," and commands that can operate flexibly on any of those objects and properties to allow a scripter to create complex statements such as "make new picture with properties {color:red}", "get the color of the first picture" and "set the color of picture 3 to red." The objects supported by an application must correspond to all of the conceptual structures of the application (windows, pictures and so on; not its internal programmatic objects). 5 points should be withheld if the application does not support the filter reference form ("whose" clauses).

Recordable — supports recording — 0 to 5 points

The application automatically generates the text of valid AppleScript statements when actions are recorded by a script editor. To receive the full 5 points, the generated script must be executable without editing when run under the same conditions under which it was recorded, and its statements should be in general form (for example, they should not refer to objects by unique id).

Attachable — supports a "Scripts" menu — 0 to 5 points

The application has a "Scripts" menu which lists AppleScripts written by the user and allows them to be run from the menu bar. An application typically installs the scripts in the Scripts menu when the user places them in a folder within the application's folder. Withhold 1 point if the Scripts menu does not support hierarchical menus (scripts installed in subfolders of the Scripts folder), and withhold 1 point if the Scripts menu is not updated "on the fly" when a new script is placed in the Scripts folder. Withhold 2 points if the menu has a name other than "Scripts" or its synonym in non-English systems.

Embeddable — supports embedding of scripts in the user interface generally — 0 to 5 points

The application provides a means to integrate user-written AppleScripts into the user interface in any of a variety of ways in addition to or instead of installing them in a "Scripts" menu. For example, scripts can be executed by clicking on buttons on a tool palette, or scripts can be run as background agents in response to events occurring in the application such as the launching of the application, the opening or closing of a window or the receipt of e-mail. The specific user interface and the manner in which these scripts are stored is not specified and is not relevant to the score. To receive the full 5 points, the manner in which embedded scripts are made available to the user should be creative, consistent with the user interface generally and easy to use.

Phase 2 — Quality of Scripting Implementation

In addition to the points awarded in Phase 1 for specific scripting features, additional points are awarded in Phase 2 on the basis of the following qualitative assessments of specific aspects of an application's AppleScript support. In general, these categories rate the degree to which an application's AppleScript support forms a consistent and useful user interface in its own right.

An award of 0 points in any one category means that the implementation is abysmal in that respect; 2 or 3 points, that it is barely useful; 6 points, that it is average and therefore acceptable; 9 or 10 points, that it is very good; and 12 points, that it is nearly perfect. In general, a conservative approach is taken so that room will be left at the extreme ends of the scale to distinguish the worst and the best implementations. An average score is not bad, and the scores of most applications should cluster around 6. A score of 4 or 5 points will not be uncommon for slightly below average implementations, and 7 or 8 points should be considered appropriate for good implementations. The maximum score for quality is 60 points. To be awarded the full score, an application's AppleScript support should achieve all of the desired qualities in each category described here.

Design — fidelity to the application's structure and function — 0 to 12 points

The Application's commands, objects and properties, and the containment hierarchy and relationship of objects to one another, flow naturally from its subject matter and function instead of rigidly and slavishly mimicking its menu commands and dialogs. The richness of the application is reflected in a relatively large number of objects, or nouns, and a relatively small number of events, or verbs, that act upon them. Commands and objects from established suites, such as the Text Suite, are implemented faithfully if the application is one for which the suite is appropriate. For an application in an area that has no established suites, its commands and objects reflect an equally appropriate fidelity to the structure and function of the application. Consistency with good scripting implementations in the run of scriptable applications in a particular field is closely observed or, if a new and different approach is taken, it is creative, imaginative and, above all, natural and intuitive — and still in keeping with the "look and feel" of standard AppleScript. Verbs and nouns that go beyond the standard features of established and well-designed suites must extend the language in a consistent and useful manner.

Completeness — depth, breadth and thoroughness of scriptability — 0 to 12 points

The application implements a large number of commands, a wide array of objects having a deep containment structure, and a rich and extensive set of properties, exposing the application's full complexity and overall functionality through its scripting interface. The AppleScript implementation is thorough, detailed and complete, making all appropriate functions of the application available to the scripter.

Transparency — resemblance to plain English words and syntax — 0 to 12 points

AppleScripts written with the application's dictionary are made up of simple English sentences. Common words in everyday use are employed in a familiar way to describe commands, objects and properties. The application's terminology and syntax are natural and intuitive, not awkward and obscure. A traditional computer programmer would think that an AppleScript statement written using the application's dictionary was a conceptual description of a script, not its executable code. A nonprogrammer would have a good chance of getting an AppleScript statement right the first time without looking at the manual, and scripters experienced with the application would have little difficulty remembering how to script it.

Stability — absence of bugs — 0 to 12 points

The application's AppleScript implementation is bug-free, and it works according to specification.

Documentation — documentation and examples are provided — 0 to 12 points

Instructions for most or all of the application's scripting features are provided, whether in print or on disk. The documentation is clearly written, well organized and thorough. A large number of useful examples are provided. Apple Guide help (or another help facility available within the application) covers the topic of scripting. The scripting dictionary ('aete' resource) within the application has comments sufficient to remind a user familiar with the application how to use its commands and objects without the need to resort to the external documentation.

Getting Your Product Rated

Send your application (including documentation), in a StuffIt archive, to <review@script.org>. Submitting to this address will cause the terms in your vocabulary to be added to the databases (both inside and outside of Apple) that are used to analyze and maintain the AppleScript language. This also submits your product for review in the Catalog of Scriptable Products. A submission form is also available in the Catalog's web area, tentatively <http://www.mainevent.com/scorecard/>. If your application is large, has printed manuals, or is packaged in a box, instructions can be found on the submission form for shipping your product, manuals, or other materials for review.

More Resources for Implementing Scriptability

A roadmap for achieving a thoughtfully designed and well-executed scripting implementation can be found in "According To Script: Steps to Scriptablility" in *develop* issue 24.

Many of the style guidelines, conventions, tips, and tricks (including the reasons for implementing the object model) are offered in "Designing a Scripting Implementation" in *develop* issue 21, and the rest of the "According to Script" series, issues 22-26 and 29.

Details of the Apple Event Manager and OSA (Open Scripting Architecture) API are described in *Inside Macintosh: Interapplication Communication.* MT

by John Schettino

Rapid Prototyping

An Object Oriented Approach to Using FaceSpan and AppleScript to Prototype Applications

Software Development 101

Just the other day I read a posting to a Macintosh programmers news group that asked a simple but powerful question. What design methodologies do most Macintosh programmers use? As I stared at the phosphors of my monitor, I recalled my years of formal study in Software Engineering. I learned all the state of the art (in the late 80s) software design methods when I was getting my Masters degree — but what do I actually use in practice?

The answer is a bit surprising. It turns out that I usually use a technique called either incremental development or rapid prototyping. This approach relies on my own expertise in ad-hoc design and the fact that I'm usually working alone on a project. It doesn't hold up quite so well for large programmer teams. The basic idea is to quickly implement a subset of the application, use it — or better yet let the customer use it, see what works and what doesn't, and then repeat the cycle. Each time through the cycle refine:

- the requirements (what the application is supposed to do)
- the interface (how the application looks)
- the behavior (what the application does), and
- the design (how the application is structured)

For this approach to succeed, using tools that support very fast implementation of successive versions of the prototype is critical. Discarding the prototype once it has served its purpose is also critical. This prevents overattachment to the initial version's implementation. It also keeps me from over-polishing the prototype! For these reasons I've selected AppleScript and FaceSpan as my Rapid Prototyping Environment.

The Tools

AppleScript is the Macintosh's bundled scripting language. It is a reasonable programming language for prototyping, since it includes basic block-programming constructs, subroutines, and several data types. It supports classes with data and member functions, and fairly complex data structures. There is also a wide array of scripting additions on the Internet that I use to round out the language. AppleScript is close enough to Java, C, or C++ to allow me to re-implement the prototype in any of those languages. It also reads a bit like psudocode so it can be used as a specification language for the final implementation. The big plus for AppleScript is that it can call on third-parties to perform complex tasks. That means databases, emailers, text editors, and even the Finder can be called on to perform major portions of a prototyped application. The big minus with AppleScript is that it completely lacks any capabilities for interface creation.

FaceSpan is a product from Digital Technology International that works as an interface tool with AppleScript or other OSA

John Schettino is an author and Senior Member of the Technical Staff at GTE Laboratories, Inc. He is the co-author of the books BASIC for the Newton: Programming for the Newton with NS BASIC and AppleScript Applications: Building Applications with FaceSpan and AppleScript, both published by AP Professional. He is also a contributing editor for the Handheld Systems Journal and for the web eZine Mobilis, where he writes about Newton and WindowsCE. You can reach him via <http://members.aol.com/pdcjohns>.

scripting languages, such as Frontier. Using FaceSpan I can create interfaces for AppleScript prototypes. The interface consists of several different styles of windows containing standard Macintosh interface elements. I then attach scripts to the windows and elements to process the events they generate. All the windows and scripts are contained in a single project file. I run the project within the FaceSpan environment while I'm developing it. Once it is working correctly, I save the project as a standard Macintosh executable file. In many ways it's simpler than it sounds.

The combination of AppleScript and FaceSpan is similar to HyperCard with several key advantages. Most important is the elimination of the Card/Background/Stack metaphor. In FaceSpan each window and it's window items behave as a unit, with their own message hierarchy. This yields "Mac-like" prototypes instead of "HyperCard-like" prototypes. FaceSpan also fully supports color, QuickTime, Drag and Drop, and other key Mac technologies. It has a rich set of interface elements to choose from. I've done pretty fancy interfaces with FaceSpan, creating and destroying window items on the fly, implementing direct manipulation interfaces, and so on. Some of the more interesting effects require a bit of script code, but at least that code is easily reused in other projects.

THE METHOD

I like methods. A method is a little less formal than a procedure. It is a general description of the steps needed to get something done. If there is something in this method that doesn't work for you, then just work around it.

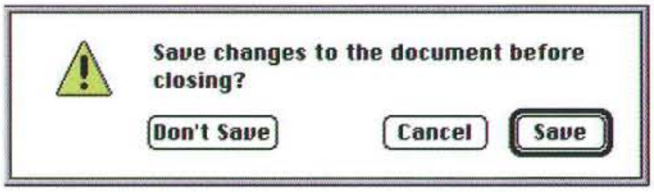
The primary reason I develop a prototype is to refine my understanding of the application I'm creating. I also use it to get end-user feedback early in the design and implementation process while it is easy to make changes. The method I use to meet these goals with FaceSpan and AppleScript is quite simple:

- First, create the static portions of the interface.
- Second, add enough AppleScript to animate the interface at the most basic level.
- Finally, add more AppleScript to implement as much of the application behavior as is needed for the prototype.

This three-step approach is not specific to FaceSpan and AppleScript. It's the same method I use in any language. When using these tools it is just a lot faster to complete each step. Depending on the goal of the prototype it is usually possible to stop before completing all three steps. For example, a UI designer might simply want to draw a possible interface and add a bit of AppleScript to animate it. The prototype could then be handed over to users and its usability could be assessed.

Drawing Interfaces in FaceSpan

Begin with whatever you know about the desired application's look and behavior and start creating windows in FaceSpan. This is a good time to review the Macintosh Human Interface Guidelines, also known as the human-computer interface (HCI) guidelines. I'm a big fan of following these, because applications that follow the guidelines are easier for users to learn and use. The FaceSpan Window Editor is a great environment to experiment with different layouts, but even more so, it's a place where the developer can see first hand the effects of following (or ignoring) the HCI guidelines. Here's a simple example. There are detailed rules for laying out the elements of a modal dialog box. The first dialog box below follows these rules to the letter, the second does not.



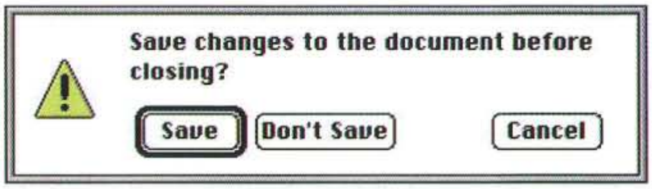


Figure 1. *Conforming and Non-conforming dialog boxes.*

Although the two contain the same buttons, the first conforms to the HCI guidelines. The default push button is in the right location, there is a correct amount of white space between the window items and the window border, and the icon is placed correctly. The result is a dialog box that is instantly recognizable to a Mac user. FaceSpan encourages this type of consistency throughout the application, but it by no means enforces it.

FaceSpan's Window Editor creates three different types of windows. Document windows are used for main windows. They can be resizable, closeable, and zoomable. Their optional titles are distinct from their name, as used in scripts. Modal windows can be titled or untitled and, unlike Document windows, must be closed before any other windows can receive events. The last window type is floating Windoid. These are windows that float on top of all Document windows. They are usually used for tool palettes.

A window contains zero or more window items. The upper limit of window items is 330. There are 14 different types of window items, and these types are further customizable using Forms. For example, the Button window item can have either a standard or a 3-D visual representation.

The Window Editor consists of a pair of tool palettes. The window under construction is displayed almost exactly as it will appear. New window items can be dragged onto the window, or drawn on the window. Window items have several properties that control their behavior. For example the visible and enabled properties determine if a particular window item is visible, and usable. If its enable property is false, it displays in a grayed-out style.

Room for Development

APS Technologies carries everything you need to store, archive, protect, backup, transport and distribute your next killer app. Listed below are just a few of our most popular products, give us a call to order a product, call for a free APS catalog; call for information... just call 1-800-761-8133. Or visit our online catalog, at www.apstech.com for literally thousands of storage solutions, peripherals and supplies. If you work in a mixed-platform environment, we've still got you covered – with Mac, PC and NT products.

APS HIGH-PERFORMANCE ULTRA SCSI DRIVES

Model	Description	EL	Int.	SR 2000	Pro
APS Q 2000	Quantum Fireball ST, 2079MB, 5.4K	$^{\$}219^{95}$	$^{\$}229^{95}$	$^{\$}299^{95}$	$^{\$}329^{95}$
APS Q 3000	Quantum Fireball ST, 3118MB, 5.4K	289^{95}	299^{95}	369^{95}	399^{95}
APS Q 4000	Quantum Fireball ST, 4136MB, 5.4K	319^{95}	339^{95}	409^{95}	439^{95}
APS Q 6400	Quantum Fireball ST, 6236MB, 5.4K	429^{95}	449^{95}	519^{95}	549^{95}
APS Q 4500	Quantum Viking, 4345MB, 7200 rpm		599^{95}	669^{95}	699^{95}
APS Q 4300	Quantum Atlas II, 4341MB, 7200 rpm		599^{95}	669^{95}	699^{95}
APS Q 9000	Quantum Atlas II, 8682MB, 7200 rpm		879^{95}	949^{95}	979^{95}
APS ST 2000	Seagate 32272N, 2157MB, 7200 rpm		Call	449^{95}	479^{95}
APS ST 4300	Seagate Barracuda, 4340MB, 7200 rpm		Call	649^{95}	679^{95}
APS ST 4500	Seagate Cheetah, 4348MB, 10000 rpm		N/A	N/A	779^{95}
APS ST 9000	Seagate Barracuda, 8683MB, 7200 rpm		Call	$1{,}019^{95}$	$1{,}049^{95}$
APS ST 9100	Seagate Cheetah, 8681MB, 10000 rpm		N/A	N/A	$1{,}199^{95}$
APS WD 2000	Western Digital Enterprise, 7200 rpm		399^{95}	469^{95}	499^{95}
APS WD 4300	Western Digital Enterprise, 7200 rpm		599^{95}	669^{95}	699^{95}

APS HIGH-PERFORMANCE ULTRA SCSI DRIVES

Model	Description	Full Height
APS ST 23000	Seagate Elite 23, 22.1GB, 5400 rpm	$^{\$}1{,}999^{95}$
APS ST 23000 W	Seagate Elite 23, 22.1GB, 5400 rpm	$2{,}099^{95}$

APS ULTRA WIDE SCSI DRIVES

Model	Description	Int.	SR 2000	Pro
APS Q 4300 W	Quantum Atlas II, 4341MB, 7200 rpm	$^{\$}619^{95}$	$^{\$}689^{95}$	$^{\$}719^{95}$
APS Q 4500 W	Quantum Viking, 4345MB, 7200 rpm	619^{95}	689^{95}	719^{95}
APS Q 9000 W	Quantum Atlas II, 8682MB, 7200 rpm	899^{95}	969^{95}	999^{95}
APS ST 2000 W	Seagate 82272W, 2157MB, 7200 rpm	Call	499^{95}	529^{95}
APS ST 4300 W	Seagate Barracuda, 4340MB, 7200 rpm	Call	699^{95}	729^{95}
APS ST 4500 W	Seagate Cheetah, 4348MB, 10,000 rpm	N/A	N/A	829^{95}
APS ST 9000 W	Seagate Barracuda, 8683MB, 7200 rpm	Call	$1{,}069^{95}$	$1{,}099^{95}$
APS ST 9100 W	Seagate Cheetah, 8681MB, 10,000 rpm	N/A	N/A	$1{,}249^{95}$
APS WD 2000 W	Western Digital Enterprise, 7200 rpm	429^{95}	499^{95}	529^{95}
APS WD 4300 W	Western Digital Enterprise, 7200 rpm	629^{95}	699^{95}	729^{95}

APS IDE DRIVES

Model	Description	Internal
APS Q 3000	Quantum Fireball ST, 3118MB, 5400 rpm	$^{\$}229^{95}$
APS Q 4000	Quantum Fireball ST, 4136MB, 5400 rpm	279^{95}
APS Q 6400	Quantum Fireball ST, 6236MB, 5400 rpm	389^{95}

APS RAID

Model	Description	Internal
APS ShortStack	Quantum Ultra SCSI 6GB/12GB	$^{\$}1{,}299^{95}$
APS Hardware RAID	Quantum Ultra SCSI w/5 - 4.3GB	$7{,}999^{95}$

APS POWERBOOK STORAGE

Model	Description	Internal
APS PowerBook Drive	IBM DMAA-21080, 1080MB, 4000 rpm	$^{\$}399^{95}$

APS REMOVABLE DRIVES

Model	Description	Int.	SR 1000	SR 2000
APS SQ 5200	SyQuest 5200, 190MB	N/A	N/A	$^{\$}389^{95}$
APS Jaz	(with 1 cartridge) 1GB	N/A	$^{\$}399^{95}$	399^{95}

APS MO DRIVES

Model	Description	SR 1000	SR 2000	Pro
APS 230 MO	(with 1 cartridge) 217MB	$^{\$}299^{95}$	$^{\$}369^{95}$	$^{\$}399^{95}$
APS 640 MO	Fujitsu M2513A2#N	N/A	469^{95}	499^{95}
APS 2.6GB MO	Sony SMO-F544, 2.4GB	N/A	$1{,}769^{95}$	$1{,}799^{95}$

APS CD-ROM DRIVES

Model	Description	External
APS CD24	24X CD-ROM in Slimline Case	$^{\$}179^{95‡}$
APS CD-R Plus	2X record/6X read CD-R (Sony)	379^{95}
APS CD-R Pro	4X record/6X read CD-R in Pro Enclosure	529^{95}
APS CD-RW	2X record/6X read CD-RW in Pro Enclosure	529^{95}
APS Jaz/CD-R	System 2X record/6X read CD-R	799^{95}
APS 2GB/CD-RW	Sys.Q 2000 Hard Drive & 2X record /6X read CD-RW	899^{95}
APS CD-R Pro	4X record/6X read CD-R in Full Height Enclosure	729^{95}

‡Available in an SR 2000 Enclosure for an additional $100.00

APS TAPE BACKUP SYSTEMS

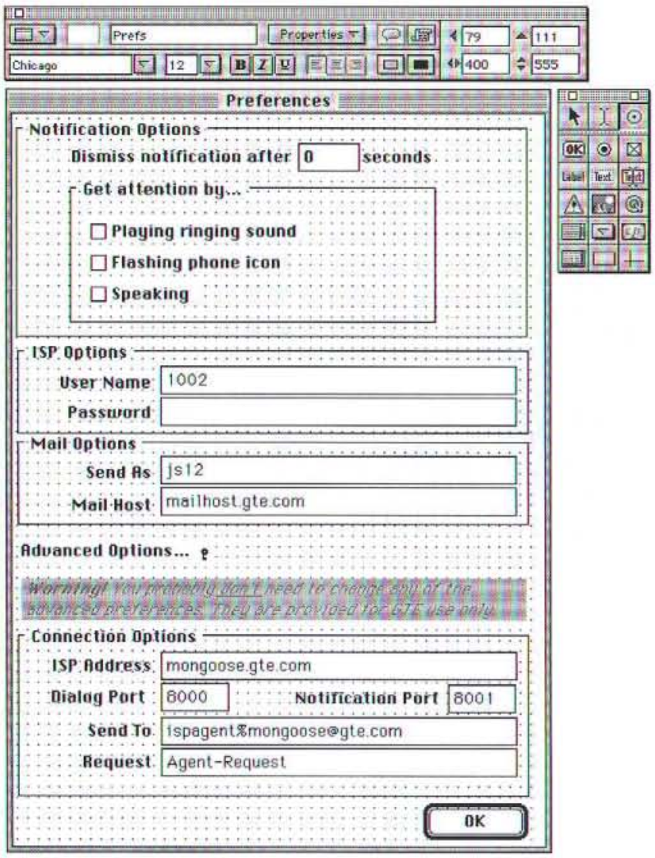
Model	Description	Internal	External
APS HyperQIC®	Travan 4 Conner QIC 3095, 8GB	$^{\$}349^{95}$	$^{\$}399^{95}$
APS HyperDAT®	DDS-2DC, 8GB	749^{95}	799^{95}
APS HyperDAT® Pro	DDS-2DC, 8GB	849^{95}	899^{95}
APS HyperDAT® III	DDS-3DC, 24GB	$1{,}149^{95}$	$1{,}199^{95}$
APS Mini Library	4mm DDS-2 AutoLoader, 64GB	N/A	$2{,}399^{95}$
APS DLT40	DLT 4000, 40GB	N/A	$2{,}999^{95}$
APS DLT70	DLT 7000, 70GB	N/A	$6{,}999^{95}$
APS AIT*	8mm W/DC, 50GB	N/A	$3{,}199^{95}$
APS DL1	4MM DDS-2 Autoloader, 136GB *Now Even Faster!*	N/A	$4{,}999^{95}$

Visit Our Online Catalog at www.apstech.com

- *30-day money-back satisfaction on all APS brand drives and accessories. Your risk is the cost of shipping.*
- *SCSI cables sold separately.*
- *Drive-for-Drive Repair or Replacement Warranty. APS will, at its discretion, replace or repair products found to be defective according to the terms of the product's warranty.*
- *Refused orders subject to 20% restocking fee.*
- *International customers must pay for all shipping charges.*
- *Listed capacities are formatted.*
- *Actual data compression and tape capacity vary greatly depending on the type of data recorded, other system parameters and environment.*
- *Prices and specifications are subject to change without notice.*
- *You need to install system software appropriate to your machine before using our hard drives.*
- *Not responsible for typographical errors.*
- *© 1997, Alliance Peripheral Systems, Inc. All Rights Reserved.*

1-800-761-8133

Technologies
APS Technologies
6131 Deramus
Kansas City, MO 64120


Prefs

Chicago 12

79 111 400 555

Preferences

Notification Options

Dismiss notification after 0 seconds

Get attention by...

☐ Playing ringing sound

☐ Flashing phone icon

☐ Speaking

ISP Options

User Name: 1002

Password:

Mail Options

Send As: js12

Mail Host: mailhost.gte.com

Advanced Options...

Warning! You probably don't need to change any of the advanced preferences. They are provided for GTE use only.

Connection Options

ISP Address: mongoose.gte.com

Dialog Port: 8000 Notification Port: 8001

Send To: ispagent%mongoose@gte.com

Request: Agent-Request

OK

Figure 2. The FaceSpan Window Editor.

Figure 2 shows a complex preferences Modal window in FaceSpan's window editor. This window uses several window items, including labels, textboxes, boxes, checkboxes, and buttons (called push buttons by FaceSpan.)

In addition to the window editor, an application has full control over the menu bar. Basic menu capabilities are supported, as well as limited (single level) hierarchical menus. The menu bar support is the weakest portion of FaceSpan. Applications can have only a single menu item in the Apple menu and have no access to the Help menu.

Animating the Interface

There are three basic forms of interface behavior: workflow control, feedback, and application specific behavior. The next step in prototyping is to implement the first two forms. By *workflow control* I mean the degree to which the application controls what the user can do at any given point in time. This includes simple things like push button and menu sensitivity, as well as window interactions. It also includes managing dependant window items. *Feedback* is the use of visual and auditory messages in the application. For example, feedback is displaying a confirmation dialog box when the user closes a window, or playing a beep when invalid data is entered into a text box.

The first step is to get the ball rolling at application startup. Let's assume I have a prototype that needs to open three windows when it is launched. I make a project in the FaceSpan environment, draw three windows in the Window Editor, and then save the project as an executable. When the user double clicks the appliction icon in the Finder, it launches just fine, but no windows are displayed. I need to add some AppleScript to the project to link the launching event to several actions. When the user launches a FaceSpan application a run event message is sent to the project script. If the user drops a file on the application icon, an open event message and a list of file references is sent to the project script instead. I want to open three windows when the user launches the application, so I add an on run handler in the project script:

```
on run
  – get prefs
  set {p1, p2, p3} to storage item "positions"
  open window "Call Manager"
    with properties {bounds:p1}
  open window "Status" ¬
    with properties {position:p2, zoomed:false}
  open window "Call Manager - driver"
    with properties {position:p3, zoomed:false}
end run
```

Most Macintosh applications keep track of window positions for the user. I can do that for my prototype as well. This handler uses a storage item to keep track of user placement of the windows. Storage items are named containers where an application can store and recall any type of data. The actual data is kept in the resource fork of the application. Recall that this is the executable version of the project file.

When the user double clicks the application icon, the on run handler of the project script is called. I begin by recalling three positions (positions are a FaceSpan datatype that specifies the x, y, width, and height of a window) from storage. I use each position to place and size each window exactly where the user left them when they last quit the prototype. I open each window use the open window command. A Window is controlled by the settings of its properties, and I can set the values of window properties in the open window command using the with properties modifier. I use this to set the initial position of each window to the stored value. I need to create the initial values for the three positions and save them in the "positions" storage item within the FaceSpan environment. I also need to somehow capture the current location and size of each window and update the storage item when the user quits the prototype.

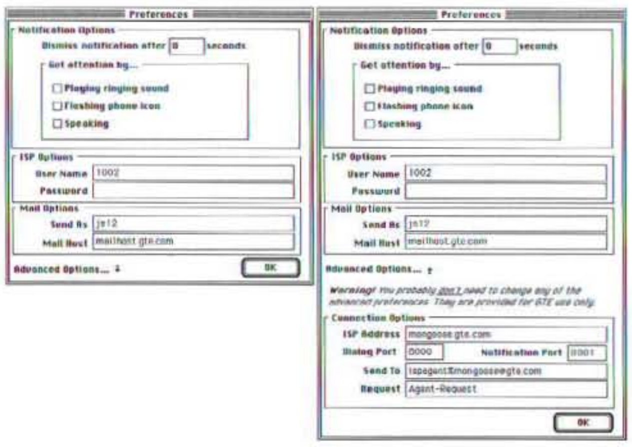
Once a window is open the user can interact with it. They can click buttons, enter text into textboxes, check checkboxes and radio buttons, and so on. If there are no scripts attached to a window then these actions don't do anything specific to the prototype. The window behaves like a fill-in-the-blanks form. All the window items and the window itself will work, but unless the prototype does something with the information or actions the user takes, not much else will. FaceSpan does a lot of the hard work: the user can enter text into text fields, select radio buttons or checkmarks, tab between fields, scroll scrollable textboxes, and what not. What is missing is the application-specific actions that link the whole interface together.

For example, the modal dialog window in **Figure 2** has a checkbox (displayed as a pull-down flag) labeled "Advanced

Options." When the user toggles this checkbox "on" the window should expand to show the window items below it, and the OK push button should move down to the bottom. Clicking it again should collapse the window back to its original size and move the OK push button back to the original location. When I draw the dialog window in the FaceSpan Window Editor I set the OK push button growth property to move both. That tells FaceSpan to move it in both directions whenever the window size changes. In effect it becomes attached to the lower right corner of the window. What I want to do is resize the window between two fixed sizes, based on the setting of the "Advanced Options" checkbox. I add a handler to the script attached to the checkbox that resizes the window when the user changes the checkbox's value. That handler is very simple:

```
on hilited theObj
  if hilite then
    set height of my window to 555
  else
    set height of my window to 345
  end if
end hilited
```

When the user clicks to push a button, checkbox, or radio button window item the hilited event is sent to its attached script. The hilite property of the checkbox is true if it is checked and false otherwise. This little script changes the window size to one of two sizes, based on the hilite property value, each time the checkbox is clicked.



Preferences
Notification Options
Dismiss notification after [0] seconds
Get attention by...
☐ Playing ringing sound
☐ Flashing phone icon
☐ Speaking
ISP Options
User Name: 1002
Password:
Mail Options
Send As: js12
Mail Host: mailhost.gte.com
Advanced Options...
OK

Preferences
Notification Options
Dismiss notification after [0] seconds
Get attention by...
☐ Playing ringing sound
☐ Flashing phone icon
☐ Speaking
ISP Options
User Name: 1002
Password:
Mail Options
Send As: js12
Mail Host: mailhost.gte.com
Advanced Options...
Warning! You probably don't need to change any of the advanced preferences. They are provided for GTE use only.
Connection Options
ISP Address: mongoose.gte.com
Dialog Port: 0000
Notification Port: 0001
Send To: ispagent%mongoose@gte.com
Request: Agent-Request
OK

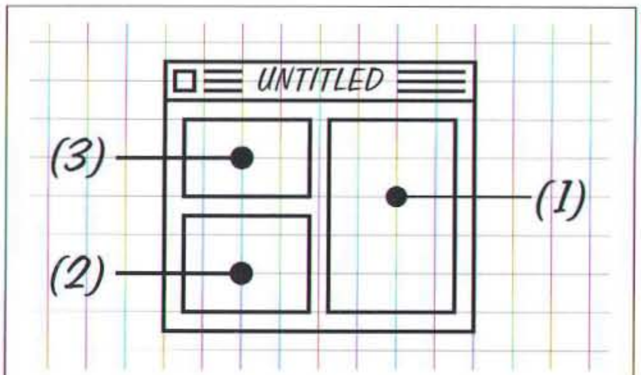
Figure 3.

I create lot of the animation of a prototype interface using similar techniques. Handlers for push button clicks (on hilited) or listbox selections (on selection made) usually enable or disable other window items, open or close other windows, or enable or disable menu items.

When the interface is fully animated the prototype is sufficient to answer a lot of questions about the final product. Does the interface work well? Is it easy to figure out how to accomplish a given task? Is feedback clear and consistent? The final step for a prototype is to add some or most of the application specific behavior. This is both a valuable and a dangerous step.

Stop reading hex.

**Save time with
Quadrivio General Edit,
the fast way to edit data files**



1 Define your file format
- Easy, familiar C-like statements.
- Complex formats handled by powerful syntax.
- Input is checked and compiled *as you type*.

2 View, edit, or analyze your data.
- Edit data *directly* in window.
- Automatically check values, at your option.
- File can be larger than RAM.

3 Multiple uses:
- Modify Finder Info and file parameters.
- Also edit data in application heaps, memory handles, and parameter RAM.
- View entire disk volumes.

Quadrivio General Edit
ADVANTAGES

✓ **Quick insight** into data structure contents.
✓ **Easier** than studying MPW's dumpfile output.
✓ **Faster debugging** with clear display of data.
✓ **Saves coding time** with direct data editing.

NEW On-line version $195

Start using General Edit just 30 minutes from now.
Visit our web site. See, Learn, and Download.

http://www.quadrivio.com

Quadrivio Corporation info@quadrivio.com
1563 Solano Avenue #360 Berkeley, CA 94707 (510)524-3246

Valuable because of what it can reveal about a potential design, and dangerous because if the prototype becomes "too functional" there is a real danger to ship it. By using FaceSpan and AppleScript, you are relieved of that danger, while still retaining the benefit of trying out a design before committing many hours to a C++ implementation. In other words, you're pretty much assured of having to throw away the prototype!

Adding Application-Specific Behavior

Adding application behavior is pretty much an exercise in AppleScript programming. The FaceSpan message hierarchy provides a lot of flexibility in the placement of handlers for FaceSpan event messages as well as application specific handlers. My job is to convert the various FaceSpan event messages into application-specific messages. FaceSpan event messages signal interface events (hilited, selection made, and so forth) while application-specific messages signal much higher-level events (produce a report, export to file, open database, and so on.) I then write handlers for those application-specific messages in AppleScript. This approach lets me reuse more of the design from the prototype because I'm able to separate the application-specific portions of each handler from the interface-specific portions. If I go on to implement the final application in C++ or Java, the interface code will be quite different than FaceSpan. The actual application specific behavior will also be coded differently, but the AppleScript code I write in the prototype can act as pseudocode.

A simple example is a push button in a window. When the user clicks the push button FaceSpan sends a hilited event message to its attached script, and then on to the window the button is in, and finally to the project script. The message flows along the message hierarchy until it is handled by a script, so I can place the on hilited handler wherever it makes the most sense. My rule is to place the handler in the script attached to the window or window item closest to the item generating the event message, unless a very similar action is performed regardless of which window item is generating it. For push buttons, I generally put the on hilited handler in the script attached to the push button. This handler converts the FaceSpan event message (hilited) into an application-specific function call. Function calls in AppleScript send a user-defined command message corresponding to the function name into the message hierarchy. Here's an example script that converts a FaceSpan event message to an application-specific user-defined command:

```
on hilited theObj
  set address to contents of textbox "digits" ¬
    as string
  sendCall(address)
  set contents of textbox "digits" to ""
end hilited
```

The handler is from a prototype Call Management application. In this prototype the main window consists of a graphical representation of a call, a textbox named "digits", and a push button labeled "Send". When the user clicks Send, I want to extract the current phone number from the textbox and place a telephone call to that number. I then want to clear the textbox. There are two different things happening here. Getting the phone number and clearing the textbox are both interface-specific actions, so I take care of them in this handler. The second action is placing a call to the entered number. That has nothing to do with the interface, and I'd like to separate the code out so I can at least have the option of reusing the design. I do this by calling a user-defined function. This in effect converts the FaceSpan hilited event message into an application-specific event message named sendCall(). Now all I have to do is implement the AppleScript handler for sendCall(). This handler is totally separate from the FaceSpan interface, and can be placed in a script anywhere along the message hierarchy (window item, window, and project). Where this is placed depends on several factors, including the ultimate target implementation language. For this prototype, I placed the handler in the project script:

```
on sendCall(phoneNum)
  if isConnected then
    tell window "Status"
      setMsg("Dial " & phoneNum)
      enableWaiting()
    end tell
    tcp write data "Dial " & phoneNum & LF ¬
      stream tcpDialogSocket
  end if
end sendCall
```

In the handler I send a protocol message through a TCP/IP socket to a host application. I also enable what amounts to a thread in another window (again using an application-specific message named enableWaiting()) that uses the FaceSpan idle handler to poll a TCP/IP socket for a reply. I'm taking advantage of the excellent shareware TCP/IP Scripting Addition by Mango Tree Software to perform the TCP/IP operations.

Don't get too hung up on the actual code, what's more important is to understand what it shows. When I implement the actual application it's pretty clear that the code must write a string to a TCP/IP socket to make a call, and once it's done this it must wait for a response. This design applies just as well to the final application as it does to this prototype.

OBJECT ORIENTED PROGRAMMING

I generally take an object-oriented approach to application design. FaceSpan and AppleScript fully support object-oriented implementations. For those not interested in or concerned with objects, a purely functional approach can also be used. If objects are important to the prototype, then there are several options available when using the FaceSpan/AppleScript combination. FaceSpan's windows and window items are themselves objects, while AppleScript includes its own script objects.

Objects in FaceSpan

In FaceSpan, every window is considered a window template. This means that FaceSpan considers every window, including all its window items and attached scripts, to be a class of sorts. An application can either use the template directly, or it can create several instances of the template. Each instance contains a complete copy of the window, including window properties and attached script properties. Each window instance is assigned a unique

If you want to be in the know, then you need every article published in the first 12 years of MacTech® Magazine and Apple's develop™ issues 1-29!

...in THINK Reference™ format!

So hurry, pick up the phone, fire up the e-mail, launch that fax machine, or simply drop by our web site and order yourself this new release of the MacTech® CD-ROM.

- Almost 1600 articles from all 139 issues of MacTech Magazine (1984-1996)
- Includes Apple develop issues 1-29
- Improved hypertext, improved indices, and a new THINK Reference Viewer — for lightning quick access!
- New hyperlinks between articles
- 100+ MB of source code — use them in your applications, with no royalties!
- Full version of THINK Reference — the original online guide to Inside Macintosh, Vols. I-VI
- 80MB of FrameWorks/SFA archives and the most complete set of FrameWorks archives known
- Sprocket™! MacTech's tiny framework that compiles quickly and supports System 7.5 features
- The best threads from the Mac programmer newsgroups plus thousands of notes, tips, snippets, and gotchas
- Popular tools that Mac programmers use to increase their productivity and much more!

Developer DEPOT™

Web Site: http//www.devdepot.com • E-mail: orders@devdepot.com
Phone: 800-MACDEV-1 • Outside the U.S. & Canada: 805-494-9797 • Fax: 805-494-9798

window id property. The open window command opens a window based on the template created in the Window Editor.

Window items also are objects. They contain properties as well as data, and can be created and destroyed dynamically. When a window is created in the Window Editor, the initial set of window items is specified and their property values are set. The application has complete control over these values, and can even override them when the window template is opened. The following code fragment dynamically creates several window items in a window.

```
on displayCall(calledNumber, calledNumberId, ¬
    isActive, isRadio, isInConf, ¬
    isConf, lineFrom, nodeX, nodeY)
  local xPos, ypos, iconArt, lineProps
  set ypos to (nodeY * 60) - top
  set xPos to nodeX * 80

  if isRadio then
    set radioItem to id of (make radio button ¬
      with properties {position:{4, ypos},width:16,¬
      height:16, script:none, hilite:isActive, ¬
      cnum:calledNumber, cid:calledNumberId} ¬
      at end of window id theWindow)
    set index of window item id radioItem to 2
  end if

  set theIndex to ¬
    (index of graphic line "divLine") + 1
  if isConf then
    set iconArt to {class:resource info, ¬
      type:"cicn", name:"group", id:5001}
  else
    set iconArt to {class:resource info, ¬
      type:"ICON", name:"Big Off Tone", id:5002}
  end if

  set iconItem to id of (make icon with properties¬
    {artwork:iconArt, position:{4 + xPos, ¬
    ypos - 10}, width:36, height:36, ¬
    draggable:true, droppable:true, script:none,¬
    selection rule:as push button, ¬
    balloon:"ISP Id is: " & calledNumberId, ¬
    cnum:calledNumber, cid:calledNumberId, ¬
    conf:isInConf} at end of window id theWindow)
  set index of window item id iconItem to theIndex

  if (lineFrom = -1) then
    set lineProps to {position:{18, ypos + 8}, ¬
      width:80, height:1, script:none}
  else if (lineFrom = 0) then
    set lineProps to {position:{(80 * (nodeX - 1)) ¬
      + 22, ypos - 60}, width:1, height:68, ¬
      script:none}
  else
    set lineProps to {position:{(80 * lineFrom) + ¬
      14, ypos + 8}, width:80 * (nodeX - lineFrom),¬
      height:1, script:none}
  end if

  set lineItem to id of (make graphic line ¬
    with properties lineProps ¬
    at end of window id theWindow)
  set index of window item id lineItem to theIndex

  if (lineFrom = 0) then
    set lineItem to id of (make graphic line ¬
      with properties {position:{(80 * (nodeX - 1))¬
      + 22, ypos + 8}, width:72, height:1,¬
      script:none} at end of window id theWindow)
    set index of window item id lineItem to theIndex
  end if

  set labelItem to id of (make label ¬
    with properties {position:{xPos + 10, ypos + ¬
    24}, width:80, height:16,contents:calledNumber,¬
    script:none} at end of window id theWindow)
  set index of window item id labelItem to theIndex

end displayCall
```

This is a meaty handler that creates several window items. It shows several interesting things. First, the algorithm used to place the radio button, icon, label, and lines is reusable. I used virtually the same algorithm in a Java and a NewtonScript version of the final application. Second, the application has total control over the placement and property values for window items it creates. Third, I'm adding application-specific properties to the radio button and icon window items. I use these window items to hold information I need (caller number and conference information) to associate with the call, but don't want to store separately. Finally, notice that each window item has its script property set to none. This is one case where I take advantage of the FaceSpan message hierarchy for performance reasons. Whenever one of these window items generates an event message I can handle it in the window's attached script with a general handler. Here's the hilited event handler:

```
on hilited theObj
  try
    if class of theObj is radio button then
      set theCallId to cid of theObj
      switchCall(theCallId)
    end if
  on error
  end try
end hilited
```

I dynamically create radio buttons, icons, graphic lines, and labels as children of the window. Of these, only radio buttons and icons generate the hilited event message. In this prototype I don't want to perform any actions if the user clicks one of the dynamic icons, and there are no other radio buttons in the window except those I create dynamically. When I add the on hilited hander to the script attached to the window, it is called whenever the user clicks something (radio button, checkbox, push button, or icon) in the window, but only if that window item does not include its own on hilited hander in its attached script. In this case I need to verify only that the class of the sender of the message is radio button. If it is, then it must be one of my dynamically created ones.

I use the class of operator to determine if the object generating the message is a radio button. If it is, then I convert the FaceSpan event message into an application specific message named switchCall(). I retrieve the application specific property cid from the radio button, since that's the id that the server process knows about. This is an example of treating window items like objects, since each item contains its own unique data.

Objects in AppleScript

AppleScript also has its own little-used object/class system. Each script in AppleScript is considered a Script object that contains methods (called handlers) and data (called properties.) The script/end script command defines a script object within a script. The basic form is:

```
script className
  property parent: otherClassName

  property p1: value1

  on handlerName(params)
  end
end script
```

A script object can have zero or one parent script property, zero or more other properties, and zero or more handlers. This is more like Java than C++ since there is only single inheritance. There is also no private or protected data or methods. Everything is public. For prototyping, this is not a major hardship.

Script objects are a little different than classes in most object-oriented languages in that the script/end script commands actually make a single instance. In the above example the commands make a single instance of the class named className. To make several instances of the class I'd have use the copy command to copy className. Instead of doing this, I can place the script/end script commands in a handler. Then the handler acts like a constructor. Every time I call the constructor a new instance of the class is created. Parameters to the hander can be used in the property statements of the script to provide initial values to newly created objects. This is an example of a simple constructor handler:

```
on makeMyObj(name)
  script myObj
    property myName: name
    on getName()
      return myName
    end getName
  end script

  return myObj
end
```

Calling makeMyObj("Fred") returns an instance of the myObj script object with its myName property initialized to "Fred".

Combining script objects with FaceSpan allows for encapsulating all of the handlers and data associated with a group of dynamic window items into one object. I use this technique to bind all the methods and data for a meeting in a meeting object in an example in by book, AppleScript Applications: Building Applications in FaceSpan and AppleScript, in the DateMinder application. This application implements a PIM that includes a Day view. The Day view is a window that displays a cluster of three window items for each meeting in a day. Here is a portion of the constructor and script object for a meeting:

```
on makeDayItem(itemclass, itemDetails, itemNote)
  local pos, startAt, meetingLength

  if itemclass = "Meeting" then
    set {pos, startAt, meetingLength} to itemDetails
    updateCalendarMeetMarks(startAt, true)
  else
    set startAt to 0
    set meetingLength to 0
    set pos to itemDetails
    updateCalendarOtherMarks(itemclass, true)
  end if

  script DayObject
    property theIndex : 0
    property barIndex : 0
    property iconIndex : 0
    property textIndex : 0
    property myPos : pos
    property myTime : startAt * 15 * minutes
    property myLength : meetingLength
    property myClass : itemclass
    property myNotes : itemNote


    on getItemInfo()
```

MkLinux

Microkernel Linux for the Power Macintosh

MkLinux is a complete system, based on Linux 2 and the Mach 3 microkernel. It includes development tools (gcc, gdb, perl, ...), X11R6.3, and hundreds of other commands.

MkLinux builds and runs most Intel-based Linux software. It works efficiently and reliably on a wide range of Power Macintosh platforms. Visit Apple Computer's MkLinux web site (www.mklinux.apple.com) and Prime Time Freeware's web site (www.ptf.com) to find out more about MkLinux!

MacPerl: Power and Ease

MacPerl is a robust and powerful port of Perl 5 to the Mac. It runs under the Finder and MPW, supports Apple Events and Toolbox calls, and is generally quite nifty! For details on MacPerl and associated products (book, CD, etc.), visit the MacPerl Pages (www.ptf.com/macperl).

Prime Time Freeware	info@ptf.com
370 Altair Way, #150	(408) 433-9662
Sunnyvale, CA 94086	(408) 433-0727 fax

```
      if itemclass = "Meeting" then
        return {itemclass:myClass, xPos:myPos,¬
          starttime:myTime, duration:myLength, ¬
          notes:myNotes}
      else
        return {itemclass:myClass, xPos:myPos, ¬
            notes:myNotes}
      end if
    end getItemInfo

    on resizeMeeting(theObj)
    end resizeMeeting

    on createImage()
    end createImage

    on updateMeeting(newTime, newDuration, newText)
    end updateMeeting

    on move (theInfo)
    end move

    on redraw()
    end redraw

    on openWindow()
    end openWindow

    on closeWindow()
    end closeWindow

    on windowName()
    end windowName

    on updateWindow()
    end updateWindow

    on adoptDetailWindow(oldWindowName)
    end adoptDetailWindow
```

New Version 7.0!

Snowbound Software

HIGH PERFORMANCE

Imaging, Annotation & Plug-in Development Tools

RasterMaster 7.0 Powerful, multi-platform raster imaging support for **60+ Raster Formats. Familiar API and well-proven technology** in use worldwide by companies such as BP, Chase Manhattan, Ford, Gannett, HP, IMSI, Kodak, LEXIS-NEXIS, Polaroid, Toyo, Unisys, Xerox *and more.*
Formats: TIF, JPEG, MO:DCA, CALS, PNG, DICOM, Flashpix, BMP, BRK, G3, G4, GIF, JED, KFX, MAC, IOCA, PCX, TGA, WFX *and more.* Features include: (* New for Version 7)

- Compression
- Image Processing
- Despeckle & Deskew
- Multipage TWAIN Scanning
- Pan, Zoom, Scroll
- Image Editing
- Auto Aspect Ratio
- Alpha Channel Support *
- CMYK 4 Plane Support *
- 10-16 bit Gray Scale *
- Anti-Aliasing in Silver *
- Fit to Width & Height *
- Tiled Image Support *
- Medical Imaging
- Animated GIF *
- Image Encryption * & more

RasterNote 2.0 Annotation/Redlining toolkit. Features include: Sticky Notes, Lines, Ellipses, Freehand Drawing, Polygons, Highlighting, Redaction, Multiple Annotation layers.

RasterNet 2.0 Plug-in for Netscape and Internet Explorer. Read 60+ Raster Formats! Simple drop In library makes it easy to enable standard browsers to read your Internet/Intranet pages.

Platforms Include: Win 95/NT, Win 3.x, Mac 68K/PPC, UNIX, Alpha, OS/2
Languages: DLL, VBX, ActiveX/OCX, Delphi, FoxPro, PB, VC++

CALL 617-630-9495 OR VISIT US AT
www.snowbnd.com Email: Salesim@snowbnd.com
Snowbound Software, 29 Crafts St. Suite 380, Newton, MA 02158 Fax: 617-630-0210

```
    on updateItemText(newNote)
    end updateItemText

    on updateItemTextAndDisplay(newNote)
    end updateItemTextAndDisplay

    on remove()
    end remove

  end script

  return DayObject

end makeDayItem
```

I call the makeDayItem() handler to create a day item object. Once it is created I draw it's graphical representation in the Day window by calling the createImage() handler. An example of this is in the handler called when the user drags an icon onto the window to create a new meeting:

```
on newDayItem(theClass, theInfo)
  local theDayItem
  set theDayItem to ¬
    makeDayItem(theClass, theInfo, "")
  set end of theDayItems to theDayItem
  tell theDayItem to createImage()
end newDayItem
```

In this handler I create a new day item object by calling the day item constructor handler, and save the object in a list of all day items stored in a property. I then call the createImage() handler in the object to draw the object in the window. Any script needing to access data associated with the object, or to change its contents or location, need call only handlers in the object. I don't have to know how it is implemented, what window items are associated with it, or how to modify their properties. As I said, it's a powerful combination.

Moving From Prototype to Product

Once the prototype is completed and ready to retire, I don't just ignore it. Rather it can be mined for valuable information. The interface will have to be reimplemented in the target language (using PowerPlant, or the Constructor for Java, or by hand coding, depending on the implementation language). The prototype serves two roles here: it acts as a guide when laying out the interface and as a specification for naming the elements of the interface.

The same holds true for the AppleScript scripts. They act as a high-level specification language for the final code. Algorithms implemented in AppleScript are highly reusable, as are data structures and class hierarchies. I'm not suggesting that it's just a matter of "porting over" to C++ or Java, but at least I've already got a working solution to use when I'm doing the final implementation.

One Example of a Prototype That Worked

This is not just some abstract article - I've used these techniques many times when creating prototypes for GTE Laboratories. Here at the labs fast turn around times are critical when we're mocking up applications. By using this method and these tools, I've created many successful demos using AppleScript and FaceSpan. When we recreate a particular application in a production system, we first capture the design by dissecting the prototype.

One such example is a Call Management client application for an Internet telephony project. This application drives a server front end to allow the user to create and manage many telephone calls. Calls can be placed, merged into conferences, and dropped. Furthermore the user can have many calls active at the same time. The ultimate implementation target for the client application was Java, but much of the protocol used to communicate between the client and server was undefined and we needed a usable prototype in just two weeks. Using the method outlined in this article, and the TCP/IP osax from Mango Tree Software, I was able to create a professional prototype with a direct manipulation interface. This prototype carried us through the demo, and then was used for the final Java application over the next two months. The running application is shown below.

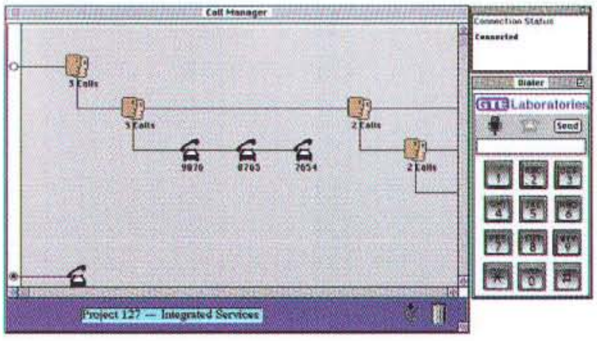


Figure 4.

As you can see, the application looks and behaves like any other Macintosh application. The icons on the scrolling display support drag & drop — just drag a phone icon on top of another to merge the calls into a conference, or drag a phone icon out of a conference and onto the main window to split that call out of the conference.

Building this prototype in FaceSpan and AppleScript let me focus on the important details including creating a layout algorithm for the call graph, finalizing and debugging the TCP/IP protocol between client and server, and creating an enjoyable and usable interface for call manipulation. I didn't need (or have time) to struggle with the details of coding the application in C++ or Java. Even so, I was still able leverage the time spent on the prototype when building the final version in Java. I also re-implemented it in a custom NewtonScript application. Even there the time spent on the prototype reaped large rewards when designing the GUI in the Newton Toolkit and implementing the application in NewtonScript.

Prototyping Can Work For You

There are at least three good reasons to create a prototype before building an application. A prototype can help answer tough questions about an interface: how usable is it, does it work better one way or another, what is the optimal placement of controls in a window. A prototype is usually ready to use in short order and can fill in for the final application while it is being written. A prototype also lets you play "what if" games with different algorithms and data structures. As long as you're willing to throw it away, it's almost always worth the investment in time.

FaceSpan and AppleScript provide a powerful and rich environment for rapid prototyping. FaceSpan interfaces can be designed such that they are full Macintosh interfaces. The tool provides a fast way to mock up interfaces, and those interfaces can be quickly modified as the prototype evolves. In a similar manner AppleScript is a wonderful prototyping language. The syntax is like psuedocode, so the resulting scripts act as documentation when moving to C++ or Java. It supports object-oriented programming, as well as a huge library of osax building blocks and access to scriptable applications. These tools and this method are valuable additions to any programmer's toolbox.

References

- John Schettino and Liz O'Hara, *AppleScript Applications: Building Applications with FaceSpan and AppleScript* AP Professional, ISBN 0-12-623957-6.
- Apple Computer, Inc., AppleScript Web Site. <http://applescript.apple.com/>.
- Apple Computer, Inc., Macintosh Human Interface Guidelines. <http://devworld.apple.com/dev/techsupport/insidemac/HIGuidelines/HIGuidelines-2.html>.
- Digital Technology International, FaceSpan Web Site. <http://www. facespan.com>.
- Mango Tree Software, "TCP/IP Scripting Addition", <http://www.mangotree.com/tcpscripadd.html>. MT

Writing scripts?
Or developing a scriptable app?

Treat yourself to

Scripter 2

and tap AppleScript's full power!

Debugging and beyond Single-step. Debug handlers individually. Try out commands in the middle of debugging. Have control over *all* your variables. Find and fix bugs on the fly!

Superior vocabulary access Explore the vocabularies of scriptable applications and scripting additions. Point-and-click to assemble commands and build object specs.

Dozens of goodies Let tons of built–in tools help you put togther your scripts. Help yourself to gobs of drag-and-drop and navigation shortcuts. Experiment with scripting commands.

Centralize your data With ScriptBase™ (included), store and retrieve values in a central place, and share values between scripts. Stash data from one application and pick it up with another.

FaceSpan integration Edit your FaceSpan scripts in Scripter with a mouseclick. And *debug* them, using all of Scripter's debugging tools.

"Scripter's virtuoso display of AppleScripting savvy and debugging wizardry make it the best environment we've found for learning or creating AppleScript scripts." ◆◆◆◆◆ **MacWeek**

BUY ONLINE **www.mainevent.com** +1 202 298 9595 1 800 616 8320

Main Event

by Rich Morin

MacPerl: A Developer's Overview

The Power of Perl, the ease of Macintosh

OVERVIEW

Perl (and, by extension, MacPerl) is a convenient and powerful language for administrative programming, CGI scripting on the World Wide Web, data analysis and filtering (such as error checking and reformatting), network programming, and more. In short, Perl can be used for almost any programming project you may have in mind.

On UNIX systems, Perl is well on its way toward taking over all substantial scripting functions, supplanting traditional tools such as sh, awk, and sed. On Macintosh systems, Perl can be used just as readily (with the added attraction that there is nothing to "unlearn").

Perl was created ten years ago when its author, Larry Wall, decided that existing scripting languages were insufficiently powerful for the distributed, bug reporting project he was working on. Seeing the potential in his new tool, Larry was gracious enough to release Perl as freeware (freely redistributable software in source and binary forms).

Other programmers picked it up, tried it, liked what they saw, and suggested enhancements and modifications. In a few years, Perl grew substantially in capabilities and in adherents, and was well on its way to becoming one of the most powerful and popular computer languages in use today. Although Perl was originally written for the UNIX operating system, it has since been ported to many different systems.

The recent rise in popularity of the World Wide Web has assured the popularity of Perl for some time to come. Perl is not a "strongly-hyped language" like Java, but it has shown itself to be an indispensible tool for creating and maintaining Web sites. Perl is used for CGI scripting, site management, and many other duties.

MacPerl (ported by Matthias Neeracher) has also been in existence for several years, but its popularity has not increased at the same rate, and certainly not to the level I feel it deserves. The Macintosh is a friendly, easy to use, and very popular computer system. MacPerl is an elegant and friendly Macintosh adaptation of an extraordinarily powerful (and popular) programming language. Why hasn't it taken off as quickly as Perl has?

Many potential MacPerl users are unaware that Perl (let alone a Macintosh version) exists! Most Macintosh magazines, rightly or wrongly, shy away from programming articles. Also, lacking any commercial reference material or distribution CD for MacPerl, many prospective users may have felt apprehensive about getting involved. PTF's MacPerl product (and articles like this one!) should resolve these issues, helping the community to grow substantially.

LANGUAGE SUMMARY

Perl syntax and fundamental capabilities are reminiscent of those found in C. The following bit of code, for instance, would work in either language:

```
printf("hello, world\n");
```

A 30-year veteran of the computer industry, **Rich Morin** <rdm@ptf.com> writes the I/Opener column for SunExpert magazine. His desktop system, Cerberus, is a three-headed Power Mac, networked to several UNIXish (FreeBSD, MkLinux, Rhapsody, Solaris, and SunOS) systems.

Rich is also the president of Prime Time Freeware <http://www.ptf.com/>, which publishes mixed-media (book/CD-ROM) collections of freely redistributable software. PTF's Mac-specific products include MacPerl: Power and Ease and MkLinux: Microkernel Linux for the Power Macintosh. This article contains material adapted from PTF's MacPerl book.

Ignoring a few dollar signs (indicating that the keyword is being used to name a scalar variable), most Perl code looks quite a bit like C code:

```
$cnt=$sum=0;
for ($i=$lo; $i<$hi; $i++) {
  if ($xyz[$i] >= 0) {
    $cnt++;
    $sum += $xyz[$i];
  }
}
printf("\$cnt=%d, \$sum=%d\n", $cnt, $sum);
```

Perl adds syntax and capabilities from several other languages, however. Here are some Perl commands that might be more familiar to an awk, sed, or shell scripter:

```
$month  =  $months{"Jan"};     # hash (associative array)
$upper  =~ tr/[a-z]/[A-Z]/;    # character substitution
$path   =~ s@/@:@g;            # regular expression
$wd     =  `pwd`;              # subprocess invocation
$cfile  =  "$base.c";          # variable interpolation
print "hello, world\n";        # unformatted print command
```

In fact, the basic Perl language offers a wealth of features unmatched in any other popular programming language, including:

- anonymous functions, defined at run-time
- arbitrary-length strings and data structures
- associative (possibly persistent) arrays (hashes)
- automatic garbage collection
- compound data structures (such as queues of arrays)
- dynamic storage allocation
- file name globbing (wild card expansion)
- late binding of data and functions
- lists: deque, queue, indexed array, stack, and more
- coercion between numbers and strings
- object-oriented features (such as inheritance)
- regular expressions for matching and substitution
- run-time evaluation of arbitrary code
- run-time tracing and control of external data
- sparse arrays (indexed and hashed)

The core language is supplemented by a wide variety of predefined objects. Some of these are included in the base distribution; others can be found on the Comprehensive Perl Archive Network (CPAN), an international set of FTP mirror sites. By looking around a little, you can find objects for arbitrary-precision arithmetic, CGI scripting, genetic sequence manipulation, network administration… well, you get the idea.

Perl's language features and object definitions work together in very powerful ways. A Perl-based CGI script can, quite trivially

- accept information from a user, checking it for validity
- retrieve data from selected files or a remote database
- perform arbitrary, user-specified calculations on the data
- generate a graph from the results, storing it as an image file
- output HTML to display the graph, with annotations

What's more, this entire operation can be performed in a demonstrably secure manner, using Perl's mechanisms for data-flow tracing, safe run-time evaluation of code, and more.

Macintosh Ease of Use

MacPerl can run as an application under the Finder or as a tool under MPW. Because most Mac users do not have MPW, the Finder version tends to dominate. I am told, however, that the MPW version acts much like any other MPW tool, supporting command-line options, ToolServer, etc.

The MacPerl application normally operates as an interactive development environment, displaying edit and interaction windows. When a MacPerl document (script) is double-clicked, it will either start up an edit/debug session or a batch program, depending on a user-definable preference. It also is possible to create "droplets", MacPerl scripts which support the Macintosh drag-and-drop protocol.

MacPerl has a built-in text editor, but it also works well with text editors such as Alpha and BBEdit. Both directly and by means of an Apple Script interface, MacPerl programs can emit and receive Apple events. Because of its interpretive nature, MacPerl provides a pleasant way to interface with the Toolbox (prototyping Dialog Boxes and such).

Although a Perl compiler is under development, current Perl implementations use a hybrid compiler-interpreter. The Perl source code is syntax-checked and parsed, but not turned into the host system's machine language. This approach allows Perl scripts to start up quickly and still run at a reasonable speed. A Perl script typically runs within a factor of three of the speed of a compiled C program.

Legal Issues

MacPerl (like Perl) is free software; it may be used, modified, and redistributed under the terms of the Perl Artistic License. This license, crafted by Larry Wall, is quite flexible. It allows commercial and non-commercial distribution of the program, with fairly minor restrictions. See the license text (included with the distribution) for specific details.

Please note that this definition of "free software" is far broader than that used by many Macintosh "freeware" applications. The fact that MacPerl is available in source code allows any interested party to look over its construction and modify its behavior. This has had a great deal to do with the growth and overall robustness of MacPerl and Perl.

Porting Issues

Perl comes from the UNIX community, which does some things rather differently than the Mac OS community. Where possible, MacPerl makes accomodations, providing "reasonable" behavior. In some cases, however, UNIX-derived code will have to be tweaked before it can be used. Finally, some kinds of Perl applications are totally unsuited for use on a Macintosh.

The Macintosh uses a carriage return (\015), rather than UNIX's line feed (\012), to separate lines of text. MacPerl accomodates this by emitting a carriage return when a newline (\n) is specified (requests for \015 are, however, taken literally). Similarly, when reading line-oriented text, MacPerl expects a carriage return, rather than a line feed.

As a result, MacPerl does not accept line feeds as delimiters in its input programs or data files. If you want to use

Have You Made a Visit to the New MacTech® Web Site?

http://www.mactech.com

Here are some highlights:

Macintosh Development News

Keep yourself informed.

The site has all the current breaking news in the development community and even archives past news pieces.

Search the Site

Search like never before.

Search and find that article, news piece or source code you're looking for via the new Phantom search engine courtesy of Maxum.

Article Archives

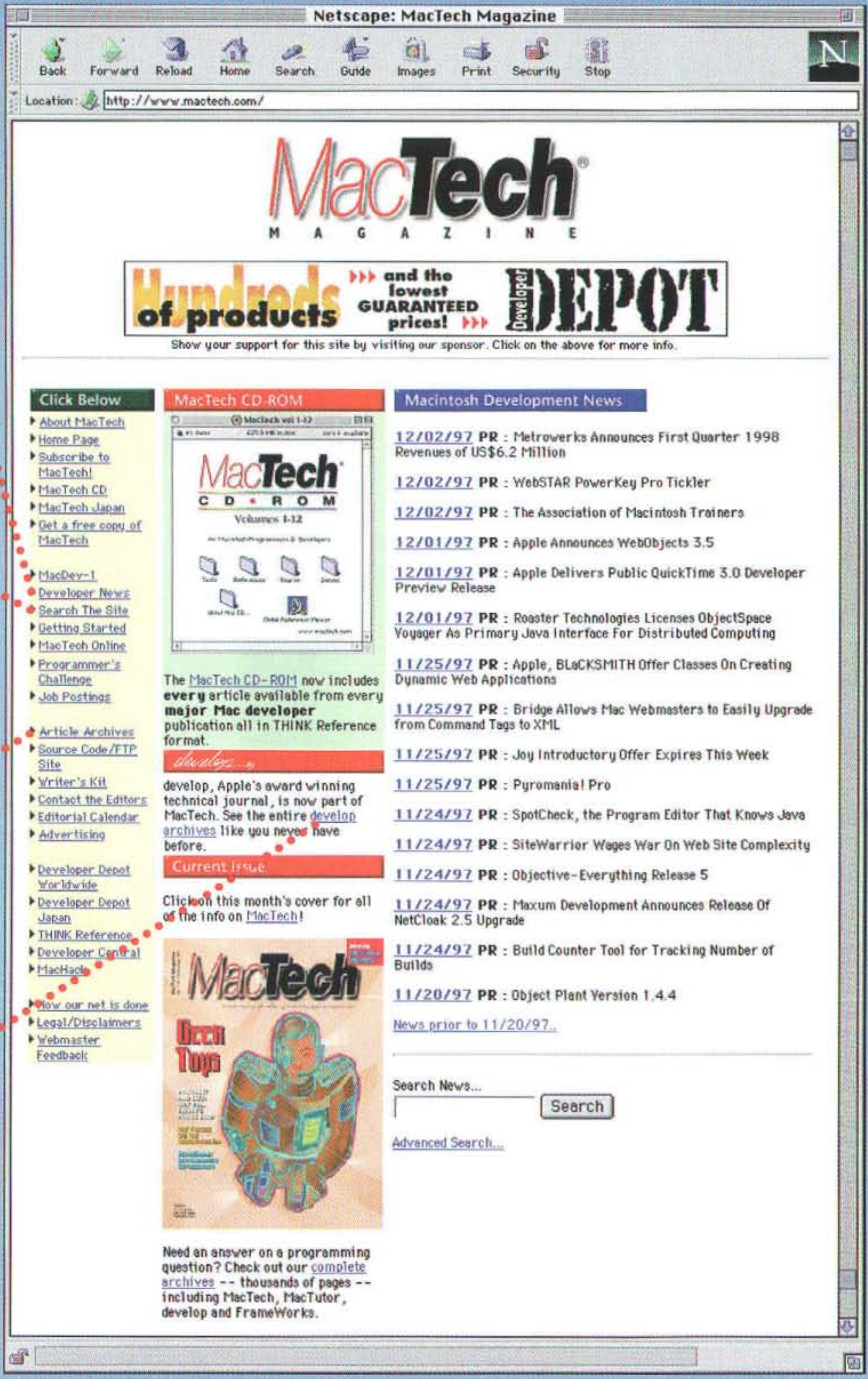
And you can have it all.

These archives have thousands of pages of content in them from the histories of MacTech, MacTutor, develop, and FrameWorks magazines.

develop

develop. Need we say more.

The entire history of develop, Apple's award winning technical journal. That is 29 issues from 1990 – 1997.



Netscape: MacTech Magazine
Back Forward Reload Home Search Guide Images Print Security Stop
Location: http://www.mactech.com/
MacTech MAGAZINE
Hundreds of products ››› and the lowest GUARANTEED prices! ››› Developer DEPOT

Show your support for this site by visiting our sponsor. Click on the above for more info.

Click Below

- About MacTech
- Home Page
- Subscribe to MacTech!
- MacTech CD
- MacTech Japan
- Get a free copy of MacTech
- MacDev-1
- Developer News
- Search The Site
- Getting Started
- MacTech Online
- Programmer's Challenge
- Job Postings
- Article Archives
- Source Code/FTP Site
- Writer's Kit
- Contact the Editors
- Editorial Calendar
- Advertising
- Developer Depot Worldwide
- Developer Depot Japan
- THINK Reference
- Developer Central
- MacHack
- How our net is done
- Legal/Disclaimers
- Webmaster Feedback

MacTech CD-ROM

The MacTech CD-ROM now includes **every** article available from every **major Mac developer** publication all in THINK Reference format.

develop

develop, Apple's award winning technical journal, is now part of MacTech. See the entire develop archives like you never have before.

Current Issue

Click on this month's cover for all of the info on MacTech!

Need an answer on a programming question? Check out our complete archives -- thousands of pages -- including MacTech, MacTutor, develop and FrameWorks.

Macintosh Development News

12/02/97 **PR** : Metrowerks Announces First Quarter 1998 Revenues of US$6.2 Million

12/02/97 **PR** : WebSTAR PowerKey Pro Tickler

12/02/97 **PR** : The Association of Macintosh Trainers

12/01/97 **PR** : Apple Announces WebObjects 3.5

12/01/97 **PR** : Apple Delivers Public QuickTime 3.0 Developer Preview Release

12/01/97 **PR** : Rooster Technologies Licenses ObjectSpace Voyager As Primary Java Interface For Distributed Computing

11/25/97 **PR** : Apple, BLaCKSMITH Offer Classes On Creating Dynamic Web Applications

11/25/97 **PR** : Bridge Allows Mac Webmasters to Easily Upgrade from Command Tags to XML

11/25/97 **PR** : Joy Introductory Offer Expires This Week

11/25/97 **PR** : Pyromania! Pro

11/24/97 **PR** : SpotCheck, the Program Editor That Knows Java

11/24/97 **PR** : SiteWarrior Wages War On Web Site Complexity

11/24/97 **PR** : Objective-Everything Release 5

11/24/97 **PR** : Maxum Development Announces Release Of NetCloak 2.5 Upgrade

11/24/97 **PR** : Build Counter Tool for Tracking Number of Builds

11/20/97 **PR** : Object Plant Version 1.4.4

News prior to 11/20/97..

Search News...

Search

Advanced Search...

a UNIX-derived Perl script or textual data file under MacPerl, you must first convert all of its carriage returns to newlines. This is trivial, if slightly annoying.

File naming syntax must also be converted. Slashes (/) must be changed into colons (:), full path names must be modified to include disk names, etc. In addition, if the UNIX code depends on "special" files (/dev/*, /proc/*, and such), some modifications will be needed.

UNIX supports preemptive multitasking, allowing (nay, encouraging) programmers to invoke separate programs whenever this seems appropriate. Perl follows in this pattern, giving programmers many ways (backquotes, exec, fork, pipes, and such) to start up other processes.

MacPerl makes a few accomodations to multitasking. Certain backquoted commands (such as `pwd`) are silently emulated, handling common UNIX idioms. Some multitasking may be performed if ToolServer is present. (Of course, the requested program also must be available!) Code which depends strongly on multitasking may not port smoothly, however, even with the aid of the ToolServer. If you find such a Perl script, you should expect to rework it quite a bit before you can use it on a Mac.

Some UNIX-derived Perl scripts will not port readily. Code which depends on multi-tasking may not port smoothly, even with the aid of ToolServer. Code that depends on UNIX-specific system calls must be modified or eliminated.

Finally, some Perl scripts rely on binary extensions which are linked into the Perl interpreter at run time. Only a few of these extensions have been ported to MacPerl, so these scripts are unlikely to work without a lot of effort (and Perl-specific knowledge).

MACPERL RESOURCES

The MacPerl Pages <http://www.ptf.com/macperl/> are a centralized source of information on MacPerl. They contain information on MacPerl, links to online resources, forms for joining the MacPerl mailing list and submitting materials (such as code samples and war stories), and more.

All modesty aside, the definitive work on MacPerl is "MacPerl: Power and Ease" (Prime Time Freeware, 1998, ISBN 1-881957-32-2, $40 MSRP). This product combines a 350+ page introductory and reference manual with a MacPerl distribution CD-ROM. For more information, visit the Prime Time Freeware web site <http://www.ptf.com/>.

Perl is served by two main web sites: <http://www.perl.com> and http://www.perl.org>. Try these sites before doing any sort of search: if you can't find what you want through one or the other of these sites, it probably doesn't exist on the Internet. Though there are quite a few books in print on Perl and related subjects, the three definitive books are published by O'Reilly & Associates:

- *Learning Perl*, Second Edition, by Randal L. Schwartz, Tom Christiansen, 1996, ISBN 1-56592-284-0.
- *Programming Perl*, Second Edition, by Larry Wall, et al, 1996, ISBN 1-56592-149-6.
- *Advanced Perl Programming* by Sriram Srinivasan, 1997, ISBN 1-56592-220-4. MT

Don't settle for less than Everything!

Objective-Everything™

- A *MUST-HAVE* tool for Rhapsody!
- The preeminent scripting solution for Yellow Box.
- Rapid prototyping and exploratory programming as you have never seen it before!
- True language independence – use the best programming language for the task at hand, or even mix-and-match languages!
- Use Python, Tcl, Perl, and Objective-C.

Objective-Browser™

- View objects in a running program.
- Investigate program structure and object inter-relations.
- Class hierarchy, methods, instance variables, and other object information can be viewed and *edited* during program execution.
- Customizable and extensible.

TipTop Software, Inc.
PO Box 30681
Bethesda MD 20824 / USA
301–656–3837, 301–656–8432 fax
tiptop@tiptop.com
www.tiptop.com

Objective-Everything, Objective-Python, Objective-Tcl, Objective-Perl, and Objective-Browser are trademarks of TipTop Software. All other trademarks belong to their respective owners.

by Steve Sheets

Contextual Menu Modules

By creating Contextual Menu Extension Modules for Mac OS 8, you can extend the behavior of all applications

A New type of Extension

The Contextual Menu (CM) Manager was introduced with Mac OS 8.0. Using this manager, developers now can provide contextual menus in their programs. When a user clicks some data while holding the control key down, a list of data specific commands can appear. For example, the Finder uses this API to display various commands (Open, Move to Trash, Get Info, Duplicate) when files & folders are control-clicked. As mentioned in last month's article, the commands that are listed come from two sources; the program and CM extension modules. This month's article will demonstrate how to create the example plug-in "CaseCharm".

At any point in a program that uses the CM API where "ContextualMenuSelect" is called, the programmer can provide 2 important pieces of information. First, he can provide the list of the commands to be displayed. Secondly, he can provide the data that is currently selected. CM extensions, also known as plug-ins, can use this data to decide what additional commands also can be added to the menu. If the user selects one of the commands that comes from the plug-in instead of from the program, the plug-in takes care of that action. The calling program is not aware that anything was selected; it just functions as if the user failed to select an item from the menu. Except for being able to look at the selected data, the plug-ins are transparent to the calling program, and the calling program is not aware of the plug-ins.

So what are CM extensions? They are PowerPC code fragments (that is, shared libraries) using the SOM Object model. Because of this, plug-ins can only run on PowerPC Macintoshes. The API is standard on Mac OS 8.0 computers, but it also can be installed on System 7.x machines. Do not assume that all the Mac OS 8.0 APIs are available for a contextual menu. Though uncommon, contextual menus can be used on a pre-8.0 machine.

CM plug-in files must have the file type of 'cmpi'. If they have the creator of 'cmnu', they will display the generic CM icon. The extension file must reside in the "Contextual Menu Items" folder in the System Folder. On Mac OS 8.0, dragging the extension onto the System Folder will automatically move the file to the correct location. On pre-Mac OS 8.0, you must drag the files manually. Once they are in the correct location, the System must be rebooted for the plug-in to be installed.

Being SOM objects, CM plug-ins are object-oriented subclasses of the super class AbstractCMPlugin. AbstractCMPlugin object has several calls that must be overridden for the extension to work. While you can create CM plug-ins that do not look at the selected data, but just provide some function, this is not a recommended. There are plenty of other way to provide commands in the Macintosh. Apple recommends that third party developers work on CM plug-ins that are data sensitive.

Steve Sheets has been happily programming the Macintosh since 1983, which makes him older than he wishes, but not as young as he acts. Being a native Californian, his developer wanderings have led him to Northern Virginia. For those interested, his non-computer interests involve his family (wife & 2 daughters), Society for Creative Anachronism (Medieval Reenactment) and Martial Arts (Fencing, Tai Chi). He is currently an independent developer, taking on any and all projects and can be reached at <MageSteve@AOL.Com>.

While the Contextual Menu Manager passes the selected data to the plug-in, the plug-in can not modify this data. At most, it can make a copy of the data, modify the copy, and then return the data by placing it in the Clipboard. The example project *CaseCharm*, discussed below, takes text data and shifts it to either upper or lower case. For example, in a word processor, a user would select a word, control-click that word, select the CaseCharm command, then paste the content of the Clipboard (the word all in upper or lower case) back into the word processor. This passing of modified data using the Clipboard is a common function of CM plug-ins.

Creating a Module Project

While a CM project can be created from scratch, the easiest way to set up one is to take an existing CM sample project and modify it for your needs. It is to easy to incorrectly set one of the required preferences in the project. In the CM Manager SDK (available from Developer CDs and websites), Apple provides a good sample program to start with. The example used in this article is based on Apple's sample code.

You must change a couple of project preferences when creating a new CM plug-in. For this project you must first change the PPC Target file name to "CaseCharm". Next, the PPC Linker Entry Point for Initialization must be changed to a new name. While it can be anything, the common usage is XXXInitialize, XXX is the name of the CM plug-in, in this case "CaseCharmInitialize".

Then you must modify the source files to match the name changes. The .cp, .h & .r files should be renamed to match the CM plug-in name. In this case, "CaseCharm.h", CaseCharm.cp" and "CaseCharm.r". Following sections will explain the changes within these files. In addition to these source files, the sample project includes the "UContextualMenuTools.cp" file. This code provides many useful routines to handle functions common to all CM plug-ins.

Additionally, various libraries must be linked in the project. These include AbstractCMPlugin, InterfaceLib, MSL RuntimePPC.Lib and SOMObjects™ for Mac OS. The file names for some of these libraries may be different for different development environments. The AppearanceLib library is required only if the CM plug-in uses the Appearance Manager.

Once all the changes are done, compile and link the code. As mentioned above, the resulting shared library must be dragged to the "Contextual Menu Items" folder and the Mac rebooted to test the code.

Adding Resources

A CM plug-in does not require much in the area of resources. Like all code fragments, the 'cfrg' resource is required. The fields containing the code fragment name & help information (in this case the strings "CaseCharm" and "A Contextual Menu Plugin to change the case of selected text") should be changed. It is always a good idea to include version information ('vers' resources) with any code.

Listing 1

CaseCharm.r

Typical resources for a Contextual Menu plug-in, in this case CaseCharm.

```
#define UseExtendedCFRGTemplate 1

#include "SysTypes.r"
#include "CodeFragmentTypes.r"

/* Code Fragment Info */
resource 'cfrg' (0) {
  { /* array memberArray: 4 elements */
    extendedEntry {
      kPowerPC,
      kFullLib,
      kNoVersionNum,
      kNoVersionNum,
      kDefaultStackSize,
      kNoAppSubFolder,
      kIsLib,
      kOnDiskFlat,
      kZeroOffset,
      kSegIDZero,
      "CaseCharm",              /* Changed for each Plug-In */
      kFragSOMClassLibrary,
      "AbstractCMPlugin",
      "",
      "",
      "A Contextual Menu Plugin to change the case of "
        "selected text."        /* Changed for each Plug-In */
    }
  }
};

/* Version Info */
resource 'vers' (1) {
  1, 0, final, 0, verUS, "1.0", "Mageware, 1997"
};

resource 'vers' (2) {
  1, 0, final, 0, verUS, "1.0", "CaseCharm 1.0"
};
```

Methods and Methods

As will be stated over and over, a CM plug-in is a SOM object. In this example, CaseCharm is a subclass of class AbstractCMPlugin. This new class overrides 4 methods of AbstractCMPlugin. CaseCharm also requires one special initialization routine for SOM to identify it. All CM plug-ins override these 4 methods (and provide an Initialization routine), so the header code is generally identical. Listing 2 shows the CaseCharm.h file. The name of the new subclass of AbstractCMPlugin must match the name given in the resource file (in this case, "CaseCharm").

Listing 2

CaseCharm.h

Typical header file for a Contextual Menu plug-in, in this case CaseCharm.

```
#pragma once
#include <AbstractCMPlugin.h>

class CaseCharm : virtual AbstractCMPlugin {

#pragma SOMReleaseOrder (Initialize, ExamineContext,
HandleSelection, PostMenuCleanup)

public:

  virtual  OSStatus Initialize(Environment* ev,
             FSSpec *inFileSpec);
  virtual  OSStatus ExamineContext(Environment* ev,
             AEDesc *inContextDescriptor,
             SInt32 inTimeOutInTicks,
             AEDescList* ioCommands,
             Boolean* outNeedMoreTime);

  virtual  OSStatus HandleSelection(Environment* ev,
             AEDesc *inContextDescriptor,
             SInt32 inCommandID);

  virtual  OSStatus PostMenuCleanup(Environment* ev);
};
```

The four methods shown here are called by the CM API when needed. Initialize() handles the initialization of the current CM. This is different from the SOM Initialization routine (defined below). It is easy to confuse the two calls. When the CM Manager is started up with the computer, a single instance of all CM extension objects is created. At that time, the Initialize() call is made. This may be before other Managers have started, so generally nothing is done in this routine. The other 3 routines are all called when a contextual menu is being created for display. The ExamineContext() method is where the plug-in adds menu items to the popup menu. This is the first time the code can examine the selected data (if any). Based on the data, the method can put any number of menu items or sub menus into the current contextual menu. The HandleSelection() call is only called if a given menu item has been selected by the user. It can then handle the actual function selected. The PostMenuCleanUp() call is the routine to be called after the current contextual menu has been handled, regardless to who handled it. The first parameter for all four of the routines is a code fragment environment parameter and is rarely used.

SOM Fragment Initializing

Being a SOM object, CaseCharm is required to have an initialization routine. The name of this routine can be anything, but it must match the name entered in the Linker preferences (CaseCharmInitialize in this case). Listing 3 shows the section of the code dealing with this call. The function calls the SOM __initialize() routine, and defines the new class. Except for the name, there is no reason to modify this code.

Listing 3 CaseCharm.cp

SOM Fragment Initializer

```
// Includes
#include <CodeFragments.h>
#include <som.xh>

// External Functions
extern pascal OSErr __initialize(CFragInitBlockPtr);

// Initializer
pascal OSErr CaseCharmInitialize(CFragInitBlockPtr init)
{
#pragma unused (init)

  OSErr theError = __initialize(init);

  if (theError == noErr)
    somNewClass(CaseCharm);

  return theError;
}
```

Setup and Shutdown

The Initialize() routine is called at startup of the current instance of a contextual menu, and is surprisingly useless. Memory and resources should be allocated when the contextual menu is being used and not kept around all the times. The PostMenuCleanUp() call can take care of anything started by ExamineContext(). Remember the HandleSelection() method may not be called if the user didn't select an items. It is more common, as in the CaseCharm example, for them to do nothing but return a noErr result.

Listing 4 CaseCharm.cp

Setup & Shutdown

```
// CM Initialize
OSStatus  CaseCharm::Initialize(Environment* ev,
            FSSpec* inFileSpec)
{
#pragma unused (ev, inFileSpec)

  return noErr;
}

// CM Clean Up
OSStatus CaseCharm::PostMenuCleanup(Environment* ev)
{
#pragma unused (ev)

  return noErr;
}
```

Coercing Text Data

Both the ExamineContext() and the HandleSelection() call are passed inContextDescriptor, a pointer to an AEDesc. This descriptor contains the selected data that the user control-clicked. This CM extension can use this descriptor to examine the selected data. UContextualMenuTools has a few routines that provide commonly used requests. Listing 5 show G_ContextualMenuTools_Has_Text_Data() (which checks if text data has been passed), G_ContextualMenuTools_Get_Text_Hdl() (which returns a copy of text passed as handle) and G_ContextualMenuTools_Get_Text_Str() (which returns a copy of text passed as string).

Since AECoerceDesc is being used, the Apple Event Manager tries very hard to return text data, even if the original data was not exactly text. For example, control-clicking a file from the finder would pass a file reference. However, AECoerceDesc would successfully convert the name of the file into text, and return that text. This is not dangerous, just something that should be noted.

Listing 5 UContextualMenuTools.cp

Handling Text Data

```
// Descriptor has text data stored in it
Boolean G_ContextualMenuTools_Has_Text_Data
          (AEDesc* p_context_descriptor_ptr)
{
  Boolean a_flag = false;

  AEDesc a_text_desc = { typeNull, NULL };
  if (AECoerceDesc(p_context_descriptor_ptr, typeChar,
        &a_text_desc) == noErr)
```

```
    if (a_text_desc.descriptorType==typeChar)
      if (a_text_desc.dataHandle) {
        long a_size = GetHandleSize(a_text_desc.dataHandle);

        a_flag = (a_size>0);
      }

  AEDisposeDesc(&a_text_desc);

  return a_flag;
}
```

// Get Text Data from descriptor as handle

```
Handle G_ContextualMenuTools_Get_Text_Hdl
          (AEDesc* p_context_descriptor_ptr)
{
  AEDesc a_text_desc = { typeNull, NULL };
  if (AECoerceDesc(p_context_descriptor_ptr, typeChar,
        &a_text_desc) == noErr)
    if (a_text_desc.descriptorType==typeChar)
      return a_text_desc.dataHandle;

  AEDisposeDesc(&a_text_desc);

  return NULL;
}
```

// Get Text Data from descriptor as string

```
void G_ContextualMenuTools_Get_Text_Str
        (AEDesc* p_context_descriptor_ptr, Str255 p_str)
{
  p_str[0] = 0;

  Handle a_handle =
G_ContextualMenuTools_Get_Text_Hdl(p_context_descriptor_ptr);
  if (a_handle) {
    HLock(a_handle);

    short a_len = GetHandleSize(a_handle);
    if (a_len>255)
      a_len = 255;

    BlockMove(*a_handle, &p_str[1], a_len);
    p_str[0] = a_len;

    HUnlock(a_handle);
    DisposeHandle(a_handle);
  }
}
```

ADDING MENUS & SUBMENUS

Once the CM plug-in has identified that it was passed data that it can use, the code must add items and submenus to the contextual menu being created. Not surprisingly, the CM Manager uses ioCommands (an AEDesc) to do this. Each menu item requires a string and a ID number. The ID number is strictly unique for this CM plug-in. The CM Manager keeps track of ID between different plug-ins. Assuming the user selects one of the items this CM plug-in created, the HandleSelection() call will be passed that ID.

Listing 6 shows a number of routines that are useful for creating these menu items, including a routine to create a menu item: G_ContextualMenuTools_Add_MenuItem(), a routine to create a blank (dotted) menu item: G_ContextualMenuTools_Add_Blank_MenuItem(), a routine to start a sub menu: G_ContextualMenuTools_Start_SubMenu() and a routine to finish creating a sub menu: G_ContextualMenuTools_Finish_SubMenu().

Mac OS **SERIAL EXPANSION BOARDS**
PARALLEL EXPANSION BOARDS

- **2/4 PORT SCSI DEVICES**
- **2/4 PORT NUBUS SERIAL BOARDS**
- **4 PORT PCI SERIAL BOARDS**
- **NUBUS & PCI PARALLEL BOARDS**

BOARDS WORK WITH SERIAL SCANNERS, GRAPHIC TABLETS, LABEL WRITERS, MODEMS, ISDN MODEMS, AND OTHER COM TOOLBOX SAVVY HARDWARE AND SOFTWARE PRODUCTS.

TO ORDER CALL: 800-367-8465

7509 Connelley Drive • Suite D • Hanover, MD 21076

www.creative-solutions-inc.com
creatsolns@aol.com

MAC OS IS A REGISTERED TRADEMARK OF ITS RESPECTIVE OWNER.

Creative Solutions, Inc.

Listing 6

UContextualMenuTools.cp

Building Menus

// Add Menu Item to AEDescList

```
OSStatus G_ContextualMenuTools_Add_MenuItem
            (Str255 p_name, long p_command,
            AEDescList* p_command_list_ptr)
{
  OSStatus a_err = noErr;

  AERecord a_command_record = { typeNull, NULL };
  a_err = AECreateList(NULL, 0, true, &a_command_record);

  if (a_err==noErr) {
    a_err = AEPutKeyPtr(&a_command_record, keyAEName,
              typeChar, &p_name[1], p_name[0]);

    if (a_err==noErr)
      a_err = AEPutKeyPtr(&a_command_record,
                keyContextualMenuCommandID, typeLongInteger,
                &p_command, sizeof (p_command));

    if (a_err==noErr)
      a_err = AEPutDesc
                (p_command_list_ptr, 0, &a_command_record);
    AEDisposeDesc(&a_command_record);
  }

  return a_err;
}
```

// Add Blank Menu Item

```
OSStatus G_ContextualMenuTools_Add_Blank_MenuItem
            (AEDescList* p_command_list_ptr)
{
  return G_ContextualMenuTools_Add_MenuItem
            ("\p-", 0, p_command_list_ptr);
}
```

// Start making a SubMenu

```
OSStatus G_ContextualMenuTools_Start_SubMenu
```

```
          (AEDescList* p_submenu_command_list_ptr)
{
  p_submenu_command_list_ptr->descriptorType = typeNull;
  p_submenu_command_list_ptr->dataHandle = NULL;

  return AECreateList
          (NULL, 0, false, p_submenu_command_list_ptr);
}

// Finish making a SubMenu
OSStatus G_ContextualMenuTools_Finish_SubMenu
            (AEDescList* p_command_list_ptr,
             AEDescList* p_submenu_command_list_ptr,
             Str255 p_supermenu_name)
{
  OSStatus a_err = noErr;

  AERecord p_supermenu_command_list = { typeNull, NULL };

  a_err = AECreateList
            (NULL, 0, true, &p_supermenu_command_list);

  if (a_err==noErr)
    a_err = AEPutKeyPtr(&p_supermenu_command_list, keyAEName,
              typeChar, &p_supermenu_name[1],
              p_supermenu_name[0]);

  if (a_err==noErr)
    a_err = AEPutKeyDesc(&p_supermenu_command_list,
              keyContextualMenuSubmenu,
              p_submenu_command_list_ptr);

  if (a_err==noErr)
    a_err = AEPutDesc(p_command_list_ptr, 0,
              &p_supermenu_command_list);

  AEDisposeDesc(p_submenu_command_list_ptr);
  AEDisposeDesc(&p_supermenu_command_list);

  return a_err;
}
```

The add menu item function is passed the name of the menu item, the ID, and the AEDesc to attach the item to. It creates a temporary list descriptor, adds the data to it, and puts the list into the passed AEDesc. The temporary list is then disposed of. The function G_ContextualMenuTools_Add_Blank_MenuItem() calls the function G_ContextualMenuTools_Add_MenuItem() with the correct values to create such a blank menu item. When the ioCommands parameter is used with these routines, the menu item is put on the top-most menu. However, these routines also can be used on submenus.

To create a submenu, the call G_ContextualMenuTools_Start_SubMenu() to create an AEDesc. Use this descriptor just as if it was the ioCommands parameter and add menu items to it. When all the menu items are added, used G_ContextualMenuTools_Finish_SubMenu() to attach the submenu (with name) to the ioCommands. In this manner, you can create any number of submenus. Listing 7 shows how to do this in the CaseCharm code. A submenu is created. Assuming text is available, two commands (and a blank line) are added. No matter what, the About command is added, and the entire submenu is then added to the main menu.

The only other two yet to be explained parameters for ExamineContext are inTimeOutInTicks and outNeedMoreTime. The inTimeOutInTicks value is the amount of time the CM plug-in can use to examine the selected data. If the call takes longer than this, it should return, with the outNeedMoreTime parameter set to true. This allows the CM Manager to handle creating larger or more difficult menu items. For simple contextual menus that do not access the disk (like the example), it is safe to set outNeedMoreTime to false.

Listing 7

CaseCharm.cp

CM Examine Content

```
OSStatus  CaseCharm::ExamineContext(Environment* ev,
            AEDesc *inContextDescriptor,
            SInt32 inTimeOutInTicks,
            AEDescList* ioCommands,
            Boolean* outNeedMoreTime)
{
#pragma unused(ev, inTimeOutInTicks)

  if (inContextDescriptor != NULL) {
    AEDescList a_sub_menu_commands = { typeNull, NULL };

G_ContextualMenuTools_Start_SubMenu(&a_sub_menu_commands);

    if (G_ContextualMenuTools_Has_Text_Data
        (inContextDescriptor))
    {
      G_ContextualMenuTools_Add_MenuItem
        ("\pConvert to Uppercase", 1002,
        &a_sub_menu_commands);

      G_ContextualMenuTools_Add_MenuItem
        ("\pConvert to Lowercase", 1003,
        &a_sub_menu_commands);

      G_ContextualMenuTools_Add_Blank_MenuItem
        (&a_sub_menu_commands);
    }

    G_ContextualMenuTools_Add_MenuItem
      ("\pAbout CaseCharm…", 1001, &a_sub_menu_commands);

    G_ContextualMenuTools_Finish_SubMenu
      (ioCommands, &a_sub_menu_commands, "\pCaseCharm");
  }

  *outNeedMoreTime = false;

  return noErr;
}
```

More and more CM plug-ins are currently being created. The room on the contextual menus is starting to run out. If the CM plug-in you create as more than one or two commands, I suggest putting the entire list of commands into a submenu. This is becoming a common user interface; a submenu with the name of the product, a number of command menu items, a blank line, and the About command menu item(s).

Outputting Result through the Scrap

When a user selects a given menu item in the contextual menu, the ID is passed to the associated CM plug-in. Again, the selected data can be parsed out. Assuming some manipulation is done on it, the result can be passed back to the user in the Clipboard (that is the Scrap). Listing 8 shows some simple routines to return text (as handles or strings) to the user this way. Notice that G_ContextualMenuTools_Set_Scrap_Text_Hdl() does not dispose of the handle, so the program must do it explicitly after the routine is called.

Listing 8

UContextualMenuTools.cp

Set Clipboard Text

```
// Set Text Scrap using handle
void G_ContextualMenuTools_Set_Scrap_Text_Hdl
        (Handle p_handle)
{
  ZeroScrap();

  if (p_handle) {
    HLock(p_handle);
    PutScrap(GetHandleSize(p_handle), 'TEXT', *p_handle);
    HUnlock(p_handle);
  }
}

// Set Text Scrap using str
void G_ContextualMenuTools_Set_Scrap_Text_Str(Str255 p_str)
{
  ZeroScrap();

  if (p_str[0]) {
    PutScrap(p_str[0], 'TEXT', &p_str[1]);
  }
}
```

User Interface

Contextual menus can have user interfaces. If a user has selected an item, dialogs and alerts can be displayed. They have to be modal, and be cleaned up before the HandleSelection() routine returns. Listing 9 provides some tools for this. The function G_ContextualMenuTools_Standard_Alert() creates a standard information alert using the new Appearance Manager StandardAlert() call. Since the CM Manager may run on machine without the Appearance Manager, it uses the G_ContextualMenuTools_Get_Gestalt_Flag() to make sure the call is available.

More complex user interfaces can be created using resources stored in the CM plug-in file. However, this file is not open as a resource file when the code is called. The G_ContextualMenuTools_Open_Resfile() takes care of this. It uses FindFolder() to search for the Contextual Menu Folder. If that fails, it searches for the folder by name. (In pre-Mac OS 8 computers, the CM Installer appears to fail to set the Contextual Menu Folder to the correct ID.) Once the file is open, any type of resource can be accessed from it. Just remember to close the resource file when you are done.

Listing 9

User Interfaces — UContextualMenuTools.cp

```
// Check bit flag of Gestalt Selector
Boolean G_ContextualMenuTools_Get_Gestalt_Flag
          (OSType p_selector, short p_bit_num)
{
  long a_result;

  if (Gestalt(p_selector, &a_result) == noErr)
    return BitTst(&a_result, 31-p_bit_num);
  else
    return 0;
}

// Standard Alert
void G_ContextualMenuTools_Standard_Alert
          (Str255 p_title, Str255 p_info)
{
  if (G_ContextualMenuTools_Get_Gestalt_Flag
        (gestaltFSAttr, gestaltHasFSSpecCalls)) {
    short a_hit;
    StandardAlert(kAlertNoteAlert, p_title,
        p_info, NULL, &a_hit);
  }
}

// Open ContextualMenu Resource File
short G_ContextualMenuTools_Open_Resfile(Str255 p_name)
{
  short a_result = -1;
  OSErr a_err;

  if (p_name[0]) {
    if (G_ContextualMenuTools_Get_Gestalt_Flag
        (gestaltFSAttr, gestaltHasFSSpecCalls)
    && G_ContextualMenuTools_Get_Gestalt_Flag
        (gestaltFindFolderAttr, gestaltFindFolderPresent)) {
    long a_dir_id;
    short a_vol_ref_num;
    FSSpec a_file_spec;
    if (FindFolder(kOnSystemDisk, 'cmnu', kCreateFolder,
        &a_vol_ref_num, &a_dir_id) == noErr) {
      BlockMove(&p_name[0], &a_file_spec.name[0], 64);
      a_file_spec.vRefNum = a_vol_ref_num;
      a_file_spec.parID = a_dir_id;

      a_result = FSpOpenResFile(&a_file_spec, fsRdPerm);
    }
    else if (FindFolder(kOnSystemDisk, kSystemFolderType,
            kCreateFolder, &a_vol_ref_num, &a_dir_id)
            == noErr) {
      Str255 a_dir_name = "\p:Contextual Menu Items:";
      short a_dir_name_len = a_dir_name[0];
      short a_name_len = p_name[0];

      BlockMove(&p_name[1], &a_dir_name[a_dir_name_len+1],
          a_name_len);
      a_dir_name[0] = a_dir_name_len + a_name_len;

      a_err = FSMakeFSSpec(a_vol_ref_num, a_dir_id,
              a_dir_name, &a_file_spec);
      if (a_err==noErr)
        a_result = FSpOpenResFile(&a_file_spec, fsRdPerm);
      }
    }
  }

  return a_result;
}
```

Sample Code

The final listing, Listing 10, shows the HandleSelection() code for the example, CaseCharm. Depending on the command passed in inCommandID, it shows the About Box using the G_ContextualMenuTools_Standard_Alert() call, or it retrieves the text using the G_ContextualMenuTools_Get_Text_Hdl() call and passes it to the CaseCharmUpperLower(). That routine shifts the case, and returns it using G_ContextualMenuTools_Set_Scrap_Text_Hdl().

Listing 10

Handle Selection — UContextualMenuTools.cp

```
// Convert the given text (upper or lower)
//    and place it in scrap book.

void CaseCharmUpperLower(Handle p_text,
          Boolean p_is_upper)
{
  if (p_text) {
    long a_length = GetHandleSize(p_text);
    if (a_length>0) {
      HLock(p_text);

      char a_char;
      for (long a_count = 0; a_count<a_length; a_count++) {
        a_char = *(*p_text+a_count);

        if (p_is_upper) {
          if ((a_char>='a') && (a_char<='z'))
            *(*p_text+a_count) = a_char - 'a' + 'A';
        }
        else {
          if ((a_char>='A') && (a_char<='Z'))
            *(*p_text+a_count) = a_char - 'A' + 'a';
        }
      }

      HUnlock(p_text);

      G_ContextualMenuTools_Set_Scrap_Text_Hdl(p_text);
    }
  }
}
```

MT

by Bruce O'Neel, Laurel MD

An FTP Fetch Client in Tcl/Tk

A light introduction to this powerful, multi-platform scripting language

Overview

Tcl/Tk (pronounced "tickle tee-kay") is a scripting language written by Dr. John Ousterhout while he was a professor at the University of California at Berkeley. Tcl can either be a standalone shell where you issue commands (like those of unix or the MPW shell), or it can be a library which you embed into your compiled program and use to issue commands. Tk is an extension to Tcl which provides graphical interface Tcl commands enabling you to write event driven programs with graphical interfaces.

Tcl/Tk has been very popular in the unix world for a long time and has recently been ported to Mac OS and Win95/NT. As of version 8.0 of Tcl/Tk, the Mac OS and Win95/NT ports have a native look and feel on their respective platforms. This article is going to provide a brief overview of Tcl/Tk and then present a demonstration Tcl/Tk program to fetch files using FTP.

Tcl/Tk Overview

Why is Tcl/Tk interesting? First, it is a dynamic scripting language. At run-time your scripts are byte compiled and run. You can get the names of procedures and variables at run-time; you can define new commands and new control statements at run-time; you can load new source code at run-time; and you can extend Tcl with your own shared libraries at run-time. Second, you can produce Mac like interfaces using the native port of Tk and you can do this quickly and interactively. Think of it as rapid prototyping for the Mac in a free language. Third, you can write scripts which can be moved unchanged from Mac OS to Win95/NT and most unix variants. Finally, you can easily write extensions to Tcl in any compiled language on the Mac, and they can either call and be called by C or produce shared libraries. These extensions also can be cross-platform if written to be portable. As an example, a group of people at NASA's Goddard Space Flight Center have written an extension to Tcl which reads and writes a file format called FITS used in astronomy <ftp://legacy.gsfc.nasa.gov/software/ftools/release/other/fitstclmac-src.sit.hqx>.

There are a few notes on Tcl's syntax that will make reading the code easier. First, remember that Tcl works by string substitution and that, from your point of view, everything is a string. `$varname` means look up the value that is currently assigned to a variable and put that string in place of `$varname`. `[command arg arg]` means execute what ever is between the square brackets and substitute the value in place of `[command arg arg]`. Finally, curly braces are used around parts of code you want to execute later and defer evaluation until sometime in the future.

An FTP Client

I thought that a good demo of Tcl/Tk for the Mac would be an FTP client. Now, I didn't want to rewrite Fetch or Anarchie, but, I did want a useful example. The example program works but there are many features left for the reader to complete and the sample probably won't work unless you FTP to a unix system. One develops a lot of respect for Anarchie or Fetch when you try to repeat their author's work.

Bruce O'Neil <beoneel@macconnect.com> spends his work time working on astrophysics satellites and his spare time playing with his lovely wife and children. What time is left is devoted to his PowerBook.

So, even though this is just a simple example, what made it good for Tcl/Tk? First, it was quick and easy to write. I took about 4-6 hours to write most of the code, with a little bit of time to clean things up for publication. Second, the resulting executable is small at around 27 Kbytes and the UI is very Mac like. Third the same source worked on more than one system. I was also able to run this on a unix system pretty much unchanged for additional testing and on the unix system it looked like I was running a Motif application. Finally I wanted a GUI and TCP/IP sockets in my program and Tcl/Tk has all of this easily built in, debugged, and well documented. Plus, you can experiment interactively with your code rather than compile, link, run, crash, debug,and edit as you must normally do.

There are two downsides to Mac Tcl/Tk applications. The first is that you have to install Tcl/Tk. The small application depends on some shared libraries, but, you could avoid the need to already have installed Tcl/Tk by using the non-shared version. The second downside is that the current version requires quite a bit of memory. The default is 4mb but you might have to bump this up if your programs crash. Many crashes are caused by running out of memory.

Displaying aWindow

The first thing the user sees when they start the program is a dialog produced by the new_connection proc, listed below. The dialog looks like

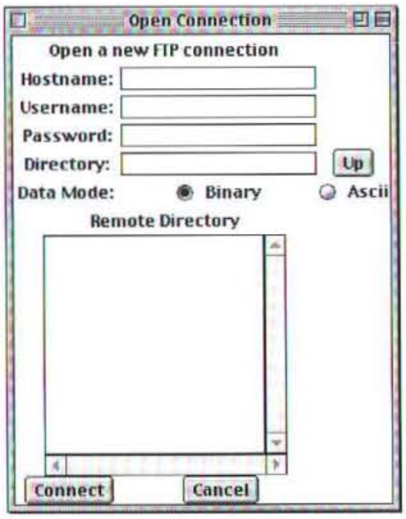


Figure 1. *Open Connection Dialog.*

Because Tcl/Tk is interactive, you could download it from <http://sunscript.sun.com> and type in each following command and watch what happens as you go. This is a very quick way to learn how Tcl/Tk works.

SPOT CHECK

THE PROGRAM EDITOR THAT REALLY KNOWS JAVA

write java that always compiles on the first try

www.genieworks.com

no more edit-compile-curse cycle

"clean up your language" with **SpotCheck**

EXAMINES YOUR JAVA CODE AS YOU EDIT EACH LINE, PROVIDING...

a immediate feedback about

parse errors
undeclared names
type conflicts and more

b smart links to name declarations

c and much more

SPOT CHECK ™

GENIE WORKS

JAVA EDITING THE WAY IT SHOULD BE

www.genieworks.com

new_connection

This is the main dialog the user interacts with and an example of Tcl/Tk programming. This asks the user for their hostname, username (optional), password (optional), and directory to connect to. When they click the connect button, it brings up a directory list of that directory.

```
# Procedure to open a new connection.
proc new_connection {} {

  # so we can access the global variable FTP
  global FTP

  # This sets the variable named t to the result of the
  # toplevel command
  # toplevel, like all Tk Widget creation commands returns
  # the name of the widget,
  # .new_connection in this case, as it's result.
  set t [toplevel .new_connection -menu .menubar]
  wm title $t "Open Connection"

  # create a text label
  label $t.title -text "Open a new FTP connection"
  # grid is a geometry manager. This puts the title on the
  # screen.
  grid $t.title -columnspan 2

  label $t.hostl -text "Hostname:"
  # associate the variable FTP(hostname) with a text entry
  # area on the screen.
  # note that there is not $ before FTP(hostname)
  entry $t.hoste -textvariable FTP(hostname)
  grid $t.hostl $t.hoste

  label $t.userl -text "Username:"
  entry $t.usere -textvariable FTP(username)
  grid $t.userl $t.usere

  label $t.passl -text "Password:"
  # -show * echos * rather than the user's keystrokes
  entry $t.passe -textvariable FTP(password) -show *
  grid $t.passl $t.passe

  label $t.dirl -text "Directory:"
  entry $t.dire -textvariable FTP(directory)

  # create a button which when it runs the command up_dir
  button $t.dirup -text "Up" -command "up_dir"
  grid $t.dirl $t.dire $t.dirup

  # put up two radio buttons to set datamode. Tied together
  # by the -variable option.
  radiobutton $t.binary -variable FTP(mode) -text Binary \
    -value Binary
  radiobutton $t.ascii -variable FTP(mode) -text Ascii \
    -value Ascii
  label $t.datamode -text "Data Mode: "
  grid $t.datamode $t.binary $t.ascii

  # frames hold things
  frame $t.direc
  label $t.direc.title -text "Remote Directory"
  # pack is another geometry manager and puts the title at
  # the top of this frame
  pack $t.direc.title -side top
  # the following three commands set up a text box and two
  # scroll bars
  set FTP(listbox) [listbox $t.direc.list \
    -xscrollcommand [list $t.direc.xscroll set] \
    -yscrollcommand [list $t.direc.yscroll set]]
  scrollbar $t.direc.xscroll -orient horizontal \
    -command [list $t.direc.list xview]
  scrollbar $t.direc.yscroll -orient vertical \
    -command [list $t.direc.list yview]
  # these pack commands put the listbox and the scrollbars on
  # the screen
  pack $t.direc.xscroll -side bottom -fill x
  pack $t.direc.yscroll -side right -fill y
  pack $t.direc.list -side left -fill both -expand true

  # put the whole frame with the remote directory listing on
  # the screen
  grid $t.direc -columnspan 2

  # attach the event of double mouse button 1 (on the Mac,
  # double click) when within
  # the widget $t.direc.list to the event of running the
  # command get_file_or_dir.
  # In other words, this sets up a routine such that when you
  # double click
  # in the list box your routine get_file_or_dir is called
  bind $t.direc.list <Double-1> {get_file_or_dir}

  button $t.connect -text Connect \
    -command "get_dir $t.direc.list"

  # destroy deletes a widget and all of it's children
  button $t.cancel -text Cancel -command "destroy $t"
  grid $t.connect $t.cancel
}
```

This code doesn't produce the nicest looking dialog, but, it's functional. It would be much prettier if I went through and added space around widgets and added colors. Note that the functions of the dialog are quite separate from the layout. This allows me to go through and change the design of the dialog without changing the supporting code.

Connecting to the Server

Once the user has filled out the connection dialog and clicked Connect it's time to get a directory listing. The bit of code which talks to the remote FTP server and gets directory looks like this:

ftp_get_dir

This bit of code reads the global FTP array variable and returns as its result the directory listing from the remote system. It connects to FTP(hostname) as user FTP(username), or anonymous if blank, using a password of FTP(password), or user@host if blank. It then changes directory to FTP(directory) and gets that directory and returns the result as a big string.

```
# The guts of getting an FTP directory. Note that this is
# the netscape connect, do
# something, and quit. Really inefficient but much easier to
# implement.
proc ftp_get_dir {} {
  global FTP
  set FTP(data_sock) 0

  update_status \
    "Getting directory from site $FTP(hostname)"

  update_status "Establishing FTP connection ..."

  # connect to the remote system
  set FTP(ftp_sock) [socket $FTP(hostname) ftp]
  fconfigure $FTP(ftp_sock) -blocking 0 -buffering none

  # call a routine ftp_read_line when the remote socket is
  # readable
  fileevent $FTP(ftp_sock) readable ftp_read_line

  if {[ftp_read] > 3} {
    return
  }

  update_status "Logging in ..."

  # send the username and password
  if {[string compare $FTP(username) ""]} {
    puts $FTP(ftp_sock) "USER $FTP(username)"
  } else {
    puts $FTP(ftp_sock) "USER anonymous"
  }
  if {[ftp_read] > 3} {
    return
  }

  if {[string compare $FTP(password) ""]} {
    puts $FTP(ftp_sock) "PASS $FTP(password)"
  } else {
```

!! Stop the press! Late-breaking News !!

We didn't have time to prepare a glitzy ad so here are "just the facts":

AppMaker

now supports Apple's Appearance controls.

These new controls — part of **Mac OS 8** and also available for **System 7** — make your program look better and also make your code simpler. They include bevel buttons, sliders, progress indicators, little arrows, tab panels, group boxes, and clock controls. They work in both windows and dialogs.

Along with the new controls are new Toolbox functions that simplify your code. Using AppMaker, it's fast and easy to create Appearance-savvy applications. AppMaker creates resources and generates source code.

Download a demo from http://members.aol.com/bowersdev. Try it out, then phone for a registration number to unlock the demo and make a full working version. Just $199, AppMaker includes a 60-day money-back guarantee and a one-year subscription on CD.

B•O•W•E•R•S
Development

P.O. Box 929, Grantham, NH 03753 • (603) 863-0945 • FAX 863-3857
bowersdev@aol.com • http://members.aol.com/bowersdev

```
  puts $FTP(ftp_sock) "PASS user@hostname"
}
if {[ftp_read] > 3} {
  return
}
```

# change to the user selected directory or /
```
if {[string compare $FTP(directory) ""]} {
  puts $FTP(ftp_sock) "CWD $FTP(directory)"
} else {
  puts $FTP(ftp_sock) "CWD /"
}
if {[ftp_read] > 3} {
  return
}

update_status "Setting up for transfer ..."
```

transfer directories in ascii mode
```
puts $FTP(ftp_sock) "TYPE A"
if {[ftp_read] > 3} {
  return
}
```

# get a server socket on our system so that the remote
system can send
# us the directory listing
```
update_status "Opening server port ..."

set serv_sock [socket -server notify_connect 0]

update_status "Setting up to retrieve directory ..."

set hostip [lindex [fconfigure $FTP(ftp_sock) -sockname] 0]
set serv_port [lindex [fconfigure $serv_sock -sockname] 2]
```

expr is how we do math
```
set serv_up [expr "int($serv_port/256)"]
set serv_lw [expr "$serv_port-$serv_up*256"]
regsub -all {\.} $hostip "," hostip
```

# send the port command to the remote system
```
puts $FTP(ftp_sock) "PORT $hostip,$serv_up,$serv_lw"
if {[ftp_read] > 3} {
  close $serv_sock
  fileevent $FTP(ftp_sock) readable ""
  close $FTP(ftp_sock)
  return
}
```

send the list command
```
puts $FTP(ftp_sock) "LIST"

if {[ftp_read] > 3} {
  close $serv_sock
  fileevent $FTP(ftp_sock) readable ""
  close $FTP(ftp_sock)
  return
}

update_status "Retrieving dir ..."

fconfigure $FTP(data_sock) -translation auto
```

# keep reading on the server socket until end of file.
```
while { ! [eof $FTP(data_sock)] } {
  set buf [read  $FTP(data_sock) 1024]
  append result $buf
}
```

# clean up and exit
```
update_status "Closing connection ..."

puts $FTP(ftp_sock) "QUIT"
fileevent $FTP(ftp_sock) readable ""
close $FTP(ftp_sock)
close $serv_sock
close $FTP(data_sock)
return $result
}
```

This bit of code talks to a remote system and implements enough of the FTP protocol to get a file listing. Basically it sends a USER command, followed by a PASS command to log in with a user name and a password. Then it sends a CWD command to change to the proper directory. Next it sends a PORT command, probably the only tricky bit. The FTP protocol uses two channels. The first is the command/result channel which is where we send commands such as USER and PASS and get responses. The second is the data channel which is where we transfer files. This is different from the http protocol where we would use the same channel for both transfers.

To request a file or directory listing from the remote system we set up a server port on the local system and tell the remote system what that port number is with the PORT command. The remote system opens a connection to that port and sends the remote file or directory listing over that connection. The PORT command has a slightly odd syntax of the form A,B,C,D,E,F where the local numeric IP address is A.B.C.D and E is the port address high byte and F is the port address low byte. Once we've gotten the port command sent, we send the LIST command. The remote system opens a socket to the port we gave it and sends the result. Once we see and end of file on our server socket we are done and can send the QUIT command. You can experiment with the FTP protocol by using a telnet client to connect to port 21 on most systems. You can also get <ftp://nic.merit.edu/documents/rfc/rfc0959.txt> and read all of the gory details.

Retrieving a file is just as easy as getting a listing. The routine ftp_get_file is almost identical to ftp_get_dir, but instead of using a LIST command to get a directory listing, we use a RETR command to get a remote file. Also, we write the file out to disk rather than returning it's contents as a string.

Adding a Menubar

Up to now all of the code has been generic Tcl/Tk. While it's nice to produce portable applications, we use Macs because we like them and we'd like our applications to look Mac-like. Tcl/Tk 8.0 has some nice features built in that we can use to make the application look more like a Mac. If we create a menu widget called say .menubar, and then add an entry to that called .menubar.apple, items on this menu will be in the Apple menu. So, we add a menubar as follows:

part of the main program

This will add the Mac menus such that they work like Mac menus. We create a menubar named .menubar and then add Apple and File entries to it. The Apple entrys will appear under the Apple menu as you'd expect and the File menu will be the first menu after the Apple menu. We'll add an accelerator to the Quit menu option with Meta-Q which will be
translated to Command-Q on the Mac.

```
# make a menubar
menu .menubar -tearoff 0

# add the file menu
.menubar add cascade -menu .menubar.file -label "File"
menu .menubar.file -tearoff 0

# add the apple menu
.menubar add cascade -menu .menubar.apple
menu .menubar.apple -tearoff 0

# add the about entry
.menubar.apple add command -label "About..." \
   -command aboutbox

# add entries to the file menu
.menubar.file add command -label "New Connection..." \
   -command new_connection
.menubar.file add separator
# this will be the normal mac quit keyboard accelerator
.menubar.file add command -label "Quit" \
   -command exit -accelerator "Meta-Q"

# make the menu the menu for the toplevel . window.Whenever
# the . window is the frontmost window then the menubar
# .menubar will be the menu at the top of the screen.

. configure -menu .menubar
```

The only other Mac specific command is console hide at the end of the program. This prevents the Tcl console from appearing. The Tcl console is where you would type Tcl commands if you were using Tcl interactively.

The last thing to do to generate a standalone Mac executable is to drag your Tcl source file onto the program Drag & Drop Tclets and answer the questions. This little program will build a Tcl executable which can be double-clicked to run our Tcl script.

Conclusion

After reading this article you should have gained an appreciation for Tcl/Tk and some things you can do with it on the Mac. It's also possible to control other programs with the TclAppleScript extension, which ships with Tcl/Tk 8.0. This allows you to use Tcl to tie together multiple programs as you can with AppleScript. Now that Tcl/Tk has native look-and-feel, the Mac Tcl scripts look like Mac programs and Tcl/Tk gives you a quick way to write Mac programs.

Bibliography and References

- Ousterhout, John K. *Tcl and the Tk Toolkit,* Addison-Wesley, 1994.
- Welch, Brent B. *Pratical programming in Tcl & Tk,* Prentice Hall, 1997.

For more information you should check the main site at <http://sunscript.sun.com/> and an excellent overview paper on Tcl/Tk and scripting languages is from <http://www.sunlabs.com/~ouster/scripting.html>. MT

by David Hempling, Bob McBeth, and Dave Mark, ©1998 by Metrowerks, Inc., all rights reserved.

CodeWarrior Latitude: Porting Custom Definition Functions and More

This month, we'll continue our ongoing look at CodeWarrior Latitude. We'll start off with a look at the issues that crop up when porting custom definition functions from the Mac to UNIX systems such as Rhapsody.

CodeWarrior Latitude is a porting library that greatly simplifies the process of porting Mac OS applications to Rhapsody. Latitude is an implementation of the Macintosh System 7 application programming interfaces (APIs) that allows source code for a Macintosh application to be compiled with little modification on a Rhapsody platform, resulting in a native Rhapsody application. Latitude maps the Mac API requests to native UNIX system calls. An application built with Latitude attains the look and feel of the new native platform.

Custom Definition Functions

Customized definition functions provided to the Mac Toolbox API as function pointers require no changes for porting from the Mac OS platform using CodeWarrior Latitude. The function pointer you provide to ModalDialog(), for example, does not need to be changed. However, you may want to add functionality to your modal definition function to handle events that weren't expected on the Mac.

On platforms supported by Latitude, modal dialog windows are moveable at all times. If the user moves a modal dialog window, the mouseDown inDrag event will go undetected in your own modal definition function ported from the Mac since on the Mac your app didn't have to deal with this possibility. Therefore, you will need to add a case for the mouseDown inDrag event, which calls DragWindow() to handle that event's processing. You will know if you neglected to handle the mouseDown inDrag event if your dialog window contains a popup menu. If DragWindow() has not been called in a moved window and the user clicks on the popup menu button, the menu will appear on screen were the popup button was on the screen before the window was moved. The DragWindow() call updates the window's port bounds which are used by PopUpMenuSelect() to display the popup menu.

Because this new possibility of unforeseen events arises on non Mac OS platforms, your own application's windows may become obscured or uncovered at times when they couldn't have been on the Mac. For example, say the user causes a system dialog to appear, such as one of Latitude's "Apple" menu desktop services like Copy Files or other system level alerts which are movable, and then the user moves one of these alerts causing a portion of your own app's windows to require updating. You now have a blank area of your application windows that you want to refresh. But, you don't have control since Latitude is servicing some other request.

Latitude provides a means for applications to register a special modal filter function whose job is to receive update events for all application windows when Latitude is in control of a presently displayed modal dialog. The filter function prototype is simply:

```
void my_app_modal_update_hook(EventRecord *theEvent)
```

You provide this function at lg_latitude_init() time as one of the field settings in the LG_APP_INFO_BLOCK:

```
void    main( long argc, char **argv )
{
#ifdef _LATITUDE_
    LG_APP_INFO_BLOCK lblock;

    lblock.flags = LG_APP_INFO_SIGNATURE |
                          LG_APP_INFO_CLASSNAME |
                          LG_APP_INFO_MODAL_EVENT_HOOK;
    lblock.signature = QUADCONST('C','T','T','R');
    lblock.classname = "Latitude";
    lblock.modal_event_hook = my_app_modal_update_hook;
    lg_latitude_init(argc,argv, &lblock);
#endif
```

Another type of custom definition function are those that define custom controls, menus, lists, and windows. Mac applications provide these functions to the Mac toolbox in one of two ways.

They either compile these functions separately into special code resources — CDEF for controls, MDEF for menus, LDEF for lists, and WDEF for windows — or they compile these functions with their application and patch special CDEF, MDEF, LDEF, or WDEF resources that contain assembly instructions. These patch resources typically contain twelve byte structures defined like this:

```
typedef struct {
  short jmp_inst;
  ProcPtr long def_func;
} DEF_STRUCT;
```

And are set up at init time like this:

```
DEF_STRUCT **my_cdef;
extern long my_cdef(short varCode, ControlHandle theControl,
        short message, long param);

my_cdef = (DEF_STRUCT **) GetResoure('CDEF', 128);

(**my_cdef).jmp_inst = 0x4EF9;
                    /* 680x0 for JMP (i.e. Jump) */
(**my_cdef).def_func = (ProcPtr) my_cdef;
```

There are two problems with this practice that come up for porting. The first problem is the 680x0 assembly code in the handle. Since CodeWarrior Latitude contains no machine code interpreter, the 680x0 JMP instruction is meaningless. The second problem is a little less obvious and far more hazardous. Since we're probably porting to a RISC based platform where long words are aligned on long word boundaries, the size of the DEF_STRUCT structure above is not 12 bytes, it's 16! (The compiler will insert 4 bytes of padding after the jmp_inst member so that def_func falls on a long word boundary.) The size of the CDEF 128 handle is 12 bytes. The assignment of the structure's def_func member writes passed the end of the 12 byte handle, thus corrupting memory.

CodeWarrior Latitude provides a means to register your definition functions so as to avoid these types of problems:

```
Handle lg_latitude_code_resource(ResType type, short id,
          StringPtr name, ProcPtr defproc);
```

Latitude creates a special handle and places it in the ROM Resource Map (ROMMapHndl). When CW Latitude's toolbox requires a CDEF, LDEF, MDEF, or WDEF, it searches the ROM Map for the registered code resource first.

If your app needs to get at the handle, you can use the one returned by lg_latitude_code_resource(), or if that one isn't readily available, you can use regular Resource Manager calls to retrieve it. The standard way to get a ROM map resource is:

```
short cur_map;
Handle h;

cur_map = CurResFile();
UseResFile(1);
h = Get1Resource(QUADCONST('C','D','E','F'), 128);
UseResFile(cur_map);
```

To get the function pointer out of a CW Latitude created code resource handle, use the call

```
ProcPtr lg_latitude_code_address(Handle h);
```

CDEFs, LDEFs and MDEFs for popup menus are completely handled by CW Latitude. MDEFs for menus off of the menubar need to be handled differently since CW Latitude is not drawing those menus itself but rather sending the appropriate instructions to the native system to display the menus.

If your app has a custom MDEF for menubar menus solely for the purpose of displaying multiple key combination menu item key equivalents (such as shift-control-option-F), then you can use a special MDEF provided by CW Latitude, called lg_latitude_custom_mdef, specifically designed for such menus.

WDEFs are also a little tricky. Again, CodeWarrior Latitude is not drawing the window itself, so you have no way of drawing a window's adornments under Latitude. However, by using some of Latitude's special window functions like lg_latitude_next_window (discussed in last month's article) and lg_latitude_register_window, you can provide Latitude with a description of the functionality of your WDEF, including floating attributes, and let Latitude handle communicating those details to the native system.

Games with A5

Macintosh applications have all kinds of games they play with the A5 register to save and restore contexts. Sometimes this is done when the application is in the background but wants to communicate something to the user. Since the native Unix systems supported by CodeWarrior Latitude do their own context switching, your application doesn't have to provide its own context management. If your application is awoken from suspension, you can be assured that the system will restore the proper stack and processor register context before executing more of your application's code.

Some applications use A5 as a sort of stack jump maneuver to recover from an error. The equivalent in Unix systems is the setjmp and longjmp pair of functions. You use setjmp to save the stack environment in a given buffer for later use by longjmp. A call to longjmp zaps execution and context back to the place were the setjmp call was made.

Games with the Memory Manager

Some Macintosh applications play interesting games with the Mac Memory Manager. They try to gobble up all of the memory available, tickling a grow zone proc in the process, to determine how much memory is really available. Others allocate a large block of memory and proceed to parse off parts of it themselves, acting as their own Memory Manager. CodeWarrior Latitude's Memory Manager provides handles and pointers the way Mac applications expect, even simulating a System Zone and an Application Zone. However there are no large, specially zoned pools of memory that Latitude allocates from.

Each handle and pointer allocation done by CW Latitude's Memory Manager is mapped directly to Unix system malloc() and realloc() calls. Since we're operating in true virtual memory systems, these common games applications play with the Memory Manager are completely meaningless. If your app is using these tricks, you should disable them and just let Latitude's Memory Manager do its thing.

Precompiled Headers

On Latitude supported systems where the available compilers do not support precompiled headers, you will have to explicitly add inclusion of your desired include files to your source code. It will probably be helpful to create one include file that sites the set of files included in the precompiled header you used on the Mac. CodeWarrior Latitude includes an include file like this for the often used MacHeaders file. It is located in the $LATITUDE_ROOT/utilities directory. You can copy it to your application's development environment for your use.

At the time of this article's writing, Metrowerks has not yet released our own C/C++/ObjC compiler for Rhapsody. However, by the time this article hits the stands, this issue of precompiled headers will be moot for Rhapsody. However, until we bring the IDE to Rhapsody, you will still have to explicitly add any includes that were implicitly added by the IDE's prefix file feature. MT

Visit MacTech® Magazine's Web site!
http://www.mactech.com

PrimeBase™
Prime Power Online

High-performance Relational DBMS
Now available for Rhapsody

Open System: ODBC, SQL, DAL Compliant
Scalable from 68K Mac to IBM RS6000
Proven Reliability (formerly P.INK SQL) with installations around the world
Cross-platform: Rhapsody, MacOS, Windows 95/NT, Sun Solaris, Linux, ...
Competitive Prices (10-user server for Apple Hardware or PC only $820)
Highly Flexible **Web Connectivity** available
EOF Adapter for OpenStep and Rhapsody
Single-user **Runtime** available

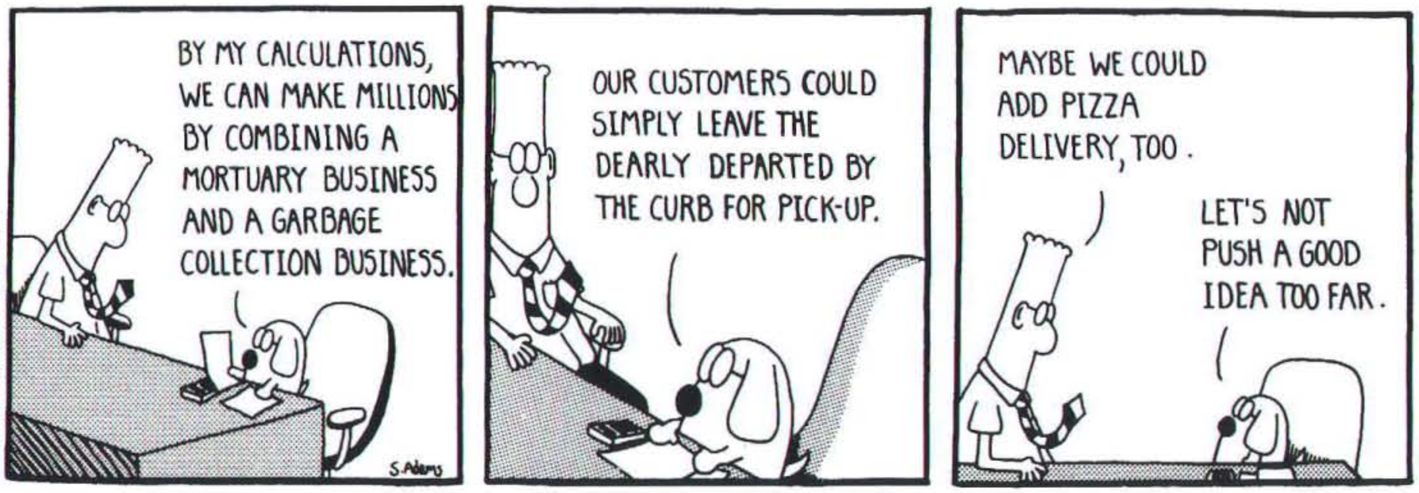
Developer Keys Free!

SNAP Innovation GmbH
Tel : +49 (40) 30636400
Fax: +49 (40) 30636333
sales@primebase.com

Download Now!
www.primebase.com

SNAP

Dilbert® *by Scott Adams*


BY MY CALCULATIONS, WE CAN MAKE MILLIONS BY COMBINING A MORTUARY BUSINESS AND A GARBAGE COLLECTION BUSINESS.
S. Adams
OUR CUSTOMERS COULD SIMPLY LEAVE THE DEARLY DEPARTED BY THE CURB FOR PICK-UP.
MAYBE WE COULD ADD PIZZA DELIVERY, TOO.
LET'S NOT PUSH A GOOD IDEA TOO FAR.

PROGRAMMER'S CHALLENGE

by Bob Boonstra, Westford, MA

IMAGE DECODER

This month's Challenge is much easier to state than it will be to implement. Your task this month is to write a decoder for Graphics Interchange Format images. GIF is a data stream-oriented file format used to define the transmission protocol of LZW-encoded bitmap data. Your code will read a file containing a GIF and draw the image in an offscreen graphics world. There are two versions of the GIF specification; this Challenge includes only the earlier and more common GIF87a format, not the GIF89a format. The format description is too long to include here, but you can find the GIF87a specification at:
<ftp://ftp.ncsa.uiuc.edu/misc/file.formats/graphics.formats/gif87a.doc>

Other useful information on graphics formats in general is available at:
<http://www.cis.ohio-state.edu/hypertext/faq/usenet/graphics/fileformats-faq/top.html>

The prototype for the code you should write is:

```
#include <QDOffscreen.h>
#include <stdio.h>
GWorldPtr ReadImage(FILE *inputFile);
```

The inputFile will have been opened for you in binary mode by the calling routine. ReadImage should read and parse the image, create a GWorld with the appropriate color table, draw the image in the GWorld, and return a GWorld pointer to the calling routine. The solution that correctly decodes a variety of images in the least amount of execution time will be the winner.

I thought for some time about the Compuserve / Unisys / LZW patent controverty before selecting this topic for the Challenge. The LZW algorithm used in GIF compression is patented by Unisys, and Unisys requires that any commercial software product that uses this algorithm, including software that reads GIF images, requires a license from Unisys. Freely distributed software does not require a license. Since the Challenge doesn't result in a software product, much less a commercial product, I decided to go ahead with this exploration of decoding efficiency.

This will be a native PowerPC Challenge, using the latest CodeWarrior environment. Solutions must be coded in C, C++, or Pascal. This Challenge was suggested by Peter Lewis, who earns 2 Challenge points for the suggestion.

THREE MONTHS AGO WINNER

Congratulations to **Jeff Mallett (Boulder Creek, CA)** for submitting the winning entry to the Pente™ Challenge. The objective was to accumulate, in minimum execution time, the most points in a round-robin tournament of the board game Pente™. With only three entries, the tournament required only six games for each player to compete against each other player twice, once playing first and once playing second. Both the winning entry and the third-place entry by Randy Boring employed an alpha-beta search technique to find the best move, and both entries were successful in winning most of their games. Jeff's entry, however, was more efficient about pruning the search tree and therefore required significantly less execution time than Randy's entry. Since a point was deducted for each second of execution time, the winning entry lost significantly fewer points for inefficiency. In addition, the scoring algorithm in Jeff's AddStone routine takes advantage of all three methods of scoring points: captures, winning by placing 5 stones in a row, and leaving sequences of 4-in-a-row on the board at the conclusion of a game. His solution earned more points as a result, even though it won the same number of games as Randy's.

THE RULES

Here's how it works: each month we present a new programming challenge. First, write some code that solves the challenge. Second, optimize your code (a lot). Then, submit your solution to MacTech Magazine. We choose a winner based on code correctness, speed, size, and elegance (in that order of importance) as well as the submission date. In the event of multiple equally desirable solutions, we'll choose one winner (with honorable mention, but no prize, given to the runner up). The prize for each month's best solution is a $100 credit for Developer Depot™.

Unless stated otherwise in the problem statement, the following rules apply: All solutions must be in ANSI compatible C or C++, or in Pascal. We disqualify entries with any assembly in them (except for challenges specifically stating otherwise.) You may call any Macintosh Toolbox routine (e.g., it doesn't matter if you use NewPtr instead of malloc). We compile all entries into native PowerPC code with compiler options set to enable all available speed optimizations. The development environment to be used for selecting the winner will be stated in the problem. **Limit your code to 60 characters per line** or compress and binhex the solution; this helps with e-mail gateways and page layout.

We publish the solution and winners for each month's Programmer's Challenge three months later. All submissions must be received by the 1st day of the month printed on the front cover of this issue.

You can get a head start on the Challenge by reading the Programmer's Challenge mailing list. It will be posted to the list on or before the 12th of the preceding month. To join, send an email to listserv@listmail.xplain.com with the subject "subscribe challenge-A".

Mark solutions "Attn: Programmer's Challenge Solution" and send it by e-mail to one of the Programmer's Challenge addresses in the "How to Communicate With Us" section on page 2 of this issue. Include the solution, all related files, and your contact info.

MacTech Magazine reserves the right to publish any solution entered in the Programmer's Challenge. Authors grant MacTech Magazine the exclusive right to publish entries without limitation upon submission of each entry. Authors retain copyrights for the code.

Onyx Technology, Inc

Spotlight Memory Debugger

NEW VERSION

Accelerated for Power Macintosh

Spotlight is the first automatic memory debugger for the Macintosh. Instantly detect invalid memory accesses, memory leaks, bad toolbox parameters, stack overwrites, memory relocation problems, and more. Spotlight can find problems in seconds that might take you a week to fix by hand.

Spotlight uses your XSYM file to automatically patch your PowerPC native application. When Spotlight detects an error it brings up the exact source code line where the error occurred along with stack crawls, memory views, and program variable usage information.

No need to change your source code. No need to recompile. No learning curve whatsoever.

Spotlight's Object Code Replacement instrumentation inspects each PowerPC processor memory read and write instruction in your code as it is run. Spotlight detects faulty memory reads and writes that occur between blocks, in released blocks, across multiple blocks, and outside of your heap.

Features

- Stops your program on exact source code line where a bad memory write occurs
- Complete leak reporting with source code stack trace showing point of allocation
- Fully validates parameters to over 400 Mac toolbox API calls.
- Works on all heap allocations Mac Toolbox, C 'malloc', C++ new and delete.
- Supports any language that produces an MPW or Metrowerks XSYM file.
- Full logging and ignore capabilities
- API for fine tuned control
- Purchase includes one year subscription to downloadable updates
- 30 day money back guarantee
- Available for secure electronic purchase now at our web site

Requirements

Macintosh or Mac OS-compatible computer with a PPC 601 or greater processor, RAM requirements vary depending on target application size; 2 MB hard disk space; System 7.5 or later; Metrowerks or MPW compatible XSYM symbolic file

Download free demo at http://www.onyx-tech.com/

$199.00

TEL	(941)795-7801	**E-MAIL**	sales@onyx-tech.com
FAX	(941) 795-5901	**URL**	http://www.onyx-tech.com/

The table below lists the number of tournament wins, points earned, cumulative score, code size, data size, and programming language for each entry. The number in parentheses after the entrant's name is the total number of Challenge points earned in all Challenges to date prior to this one.

Name	Wins	Points	Score	Code	Data	Language
Jeff Mallett (74)	3	33	32.03	4872	7923	C
Sebastian Maurer	0	10	9.97	4860	208	C++
Randy Boring (66)	3	23	-374.90	7672	2649	C

Top 20 Contestants

Here are the Top Contestants for the Programmer's Challenge. The numbers below include points awarded over the 24 most recent contests, including points earned by this month's entrants. The scores have been adjusted to correct an error made in assigning points for the Stratego Challenge printed in the November issue.

Rank	Name	Points	Rank	Name	Points
1.	Munter, Ernst	200	11.	Higgins, Charles	20
2.	Boring, Randy	73	12.	Larsson, Gustav	20
3.	Cooper, Greg	61	13.	Studer, Thomas	20
4.	Lewis, Peter	59	14.	Gundrum, Eric	15
5.	Mallett, Jeff	50	15.	Hart, Alan	14
6.	Nicolle, Ludovic	48	16.	O'Connor, Turlough	14
7.	Murphy, ACC	34	17.	Picao, Miguel Cruz	14
8.	Gregg, Xan	33	18.	Saxton, Tom	12
9.	Antoniewicz, Andy	24	19.	Cutts, Kevin	11
10.	Day, Mark	20	20.	8-way tie	10

There are three ways to earn points: (1) scoring in the top 5 of any Challenge, (2) being the first person to find a bug in a published winning solution or, (3) being the first person to suggest a Challenge that I use. The points you can win are:

1st place	20 points	5th place	2 points
2nd place	10 points	finding bug	2 points
3rd place	7 points	suggesting Challenge	2 points
4th place	4 points		

Build great applications... Better Cheaper Faster!

Great | Good'nuf

Tools Plus

LIBRARIES + FRAMEWORK

Make cool apps while others are still reading their manuals. Professionally polished. Wickedly fast. Delightfully efficient.

"...the routines are more compact and faster than anything you might write...Every element of Tools Plus is useful... a bargain compared with coding these routines yourself." MW★★★★ Macworld

REC PLAY

You build the interface.
Tools Plus provides the infrastructure.
It makes all your pieces work together as an application.
With only a few hundred routines, Tools Plus thins your code by tens of thousands of lines. You see results sooner. Changes are a snap.

"All in all, it's an incredibly rich collection of tools... If you are interested in developing applications that have 'quality' written all over them, then Tools Plus is for you." MacTech MAGAZINE

Yes, Tools Plus has it!

- **Create any element** using a single routine
- **Everything works** as soon as you create it
- **Automates** all standard GUI elements
- **Windows, Tool bar** and **floating palettes**
- **Buttons** (all kinds, flat and 3D)
- **Scroll bars** (speed control, live scrolling)
- World-class **custom controls**
- **Fields** (w/scroll bars, filters, auto-editing)
- **Picture buttons** (the best anywhere)
- **List boxes**
- **CDEF** and **LDEF** automation
- **Cursors** (color, animated, auto-change)
- **Menus** (pull-down, hierarchical, pop-up)
- **Edit menu** (undo/redo, automatic editing)
- **Panels** (3D, group boxes, lines, panes)
- **3D titles** (raised or inset)
- Extended multi-monitor and color support
- **Clipboard** automation
- **Dynamic alerts** (no resources required)
- **Event processing** automation
- Over 500K of custom fonts, icons, cursors, and other useful resources

OK OK OK

Style

Style
√ Plain
Bold
Italic
Underline
Outline
Shadow

Plus thousands more exciting features and services!

Results:
[✓ Radical]
☐ Slick
☒ Ravin'

1 2 3 4 5 6 7

Inset Panel | Raised Panel

FREE SuperCDEFs™ $89 value
✓ professionally designed and crafted controls
✓ dozens of 3D & flat buttons, tabs, sliders

Tools Plus for

Symantec (THINK) C/C++ *(68K)*	$149
THINK Pascal *(68K)*	$149
THINK C/C++ & THINK Pascal	$199
CodeWarrier Bronze *(68K)*	$199
CodeWarrier Gold *(68K and PowerPC)*	$249

(We accept VISA and Amex only. Add $10 for shipping.)
*Call for Academic pricing

Water's Edge Software
2441 Lakeshore Road West, Box 70022,
Oakville, Ontario Canada L6L 6M9

Orders and Enquiries:
Phone: (416)219-5628
Fax: (905) 847-1638

WaterEdgSW@aol.com

Free Evaluation Kit:
Available at our web site
http://www.interlog.com/~wateredg

Water's Edge Software

Here is Jeff's winning solution:

MyPente.c
Copyright © 1997 Jeff Mallett

```
//
// Do a depth=1 search to sort moves. Then do a higher
// depth alpha-beta search to select the best move.
//
// Plays for score rather than to win. For example,
// doesn't try to win by getting 5 captures since
// there is no score incentive for this. It could be
// retuned to play for a win instead.

#include "Pente.h"

#define SEARCH_DEPTH 4
enum {
  _FRIEND = 0x0001,
  _ENEMY  = 0x0002,
  _EMPTY  = 0x0004,
  _WALL   = 0x0008
};

#define BORDER          1
#define BORDERS         2
#define INFINITY        32600
#define NA               0
#define MAX_CELLS        (31 + BORDERS) * (31 + BORDERS)

#define ESTIMATE_PLUS   1
#define WIN              2000
#define CAPTURE_SCORE   240
#define VULNERABLE       150

static short CHAIN_SCORE2[3][3] = { //[color][open]
  { NA, NA, NA },
  { -1, -9, -2 },        // FRIEND
  {  1,  9,  2 }         // ENEMY
};
static short CHAIN_SCORE3[3][3] = { //[color][open]
  { NA, NA, NA },
  {  3, 16, 40 },        // FRIEND
  { -2,-10,-30 }         // ENEMY
};
static short CHAIN_SCORE4[3][3] = { //[color][open]
  {   NA,  NA,  NA },
  {  230, 240, 250 }, // FRIEND
  { -150,-155,-160 }  // ENEMY
};
static short CHAIN_SCORE5[3] = {    //[color]
  NA, WIN, -WIN
};
//                    1  2  3  4
static short BLOCK_SCORE[5] = { NA, 0, 14, 14, 22};
static short THREATS[] = {
  NA, NA,
  25,  // 2: tria
  30,  // 3: half-open four
  350, // 4: double trias
  370, // 5: tria + half-open four
  450, // 6: tessera or 2 half-open fours
  500, 500, 500, 500, 500, 500, 500, 500, 500, 500
};

static short gPreScore[MAX_CELLS], gEstimates[MAX_CELLS];
static short *gEstimatesStart, *gEstimatesEnd;
static short gBoard[MAX_CELLS], *gBoardStart, *gBoardEnd;
static short *gFirstStone, *gLastStone;
static short *gKillers[SEARCH_DEPTH], gPreviousThreats;
static short gDirections[8], *gDirectionsEnd;
static short gMoveNum, gScore, gStartDepth;
static short gBoardHalfSize, gSideLength, gBoardMax;
static short gCumCapturesFriend, gCumCapturesEnemy;
static short gAdjustment, gSE;

#define ADJUST(a)           ( (a) + gAdjustment )
#define TRANSLATE(x, y) \
  (gSideLength * ADJUST(y) + ADJUST(x))
#define GET_INDEX(p)  ((p) - gBoard)
#define GET_X(i)        ( ((i) % gSideLength) - gAdjustment )
```

```
#define GET_Y(i)          ( ((i) / gSideLength) - gAdjustment )
#define UPDATE_ENDPOINTS(pSq, a, b) \
  if (pSq < a) a = pSq; \
  if (pSq > b) b = pSq

#define OPPONENT(side)  (3 - (side))
#define OCCUPIED(x)     ((x) & (_FRIEND | _ENEMY))
#define EMPTY(x)        ((x) == _EMPTY)

// gChanges — Array of unsigned longs containing data to
// undo moves
//   list of:
//      <pointer to position> <old position value>
//   terminated by a 0L
static unsigned long gChanges[256], *gChangesEnd;
#define PUSH(x)           *(gChangesEnd++) = (x)
#define START_SAVE     PUSH(0L)
#define PUSH_SQ(pSq) \
    { PUSH((long)*(pSq)); PUSH((unsigned long)(pSq)); }
#define POP               *(--gChangesEnd)
#define TOP               *gChangesEnd

static short * ChooseNextMove();
static short AddStone(short alpha, short beta, short *pSq,
                short color, short depth,
                short capturesFriend, short capturesEnemy,
                short *firstStone, short *lastStone);
static long MyFindCaptures(Capture capture[], short *pSq,
                short us, short them);
static Boolean MyFindFive(short *pSq, short color);
```

InitPente

```
void InitPente(
  long boardHalfSize    /* e.g., 9 for a 19x19 board */
      /* all coordinates between -boardHalfSize
        and +boardHalfSize */
) {
  short i, j, *pSq;

  gBoardHalfSize = boardHalfSize;
  gSideLength = boardHalfSize * 2 + 1 + BORDERS;
  gSE = gSideLength + 1;
  gDirections[0] = -gSE;                 // NW
  gDirections[1] = -gSideLength;         // N
  gDirections[2] = -gSideLength + 1;     // NE
  gDirections[3] = -1;                   // W
  gDirections[4] = gSE;                  // SE
  gDirections[5] = gSideLength;          // S
  gDirections[6] = gSideLength - 1;      // SW
  gDirections[7] = 1;                    // E
  gDirectionsEnd = &gDirections[8];

  gBoardMax = gSideLength * gSideLength;
  i = (gSideLength + 1) * BORDER;
  gBoardStart = &gBoard[i];
  gBoardEnd = &gBoard[gBoardMax - i];
  gEstimatesStart = &gEstimates[i];
  gEstimatesEnd = &gEstimates[gBoardMax - i];
  gFirstStone = gBoardEnd;
  gLastStone = gBoardStart;

  pSq = gBoard;
  do {
    *pSq = _WALL;
  } while (++pSq != gBoardStart);
  do {
    *pSq = _EMPTY;
  } while (++pSq != gBoardEnd);
  do {
    *pSq = _WALL;
  } while (++pSq < &gBoard[gBoardMax]);
  for (i = BORDER + boardHalfSize * 2 + 1;
        i<gBoardMax-gSideLength; i += gSideLength)
    for (j=0; j<BORDERS; ++j)
      gBoard[i+j] = _WALL;
  gCumCapturesFriend = gCumCapturesEnemy = gMoveNum = 0;
  gAdjustment = gBoardHalfSize + BORDER;
  gStartDepth = SEARCH_DEPTH - 1;
}
```

Pente

```
void Pente(
  Point opponentsMove,          /* your opponent moved here */
  Boolean playingFirst,         /* ignore opponentMove */
  Point *yourMove,              /* return your move here */
  Capture claimCaptures[],      /* return captured pairs here */
  long *numCaptures,            /* return # of claimCaptures */
  Boolean *claimVictory         /* return true if you claim
                                     victory with this move */
) {
  Point move;
  Capture opponentCaptures[8];

  short *pSq, *pNewSq, *d;
  short score, bestScore, i;
  short *pBestMove;

  *numCaptures = 0;
  *claimVictory = false;
  if (playingFirst) { // *** MOVE 1
    move.h = move.v = 0;
    pSq = &gBoard[TRANSLATE(0, 0)];
    *pSq = _FRIEND;
    UPDATE_ENDPOINTS(pSq, gFirstStone, gLastStone);
    *yourMove = move;
    gMoveNum = 1;
    return;
  }

  i = TRANSLATE(opponentsMove.h, opponentsMove.v);
  pSq = &gBoard[i];
  *pSq = _ENEMY;
  UPDATE_ENDPOINTS(pSq, gFirstStone, gLastStone);
  ++gMoveNum;

  if (gMoveNum == 1) { // *** MOVE 2
    move.h = move.v = -2;
    pSq = &gBoard[TRANSLATE(-2, -2)];
    *pSq = _FRIEND;
    UPDATE_ENDPOINTS(pSq, gFirstStone, gLastStone);
```

TestTrack

Your Total Bug Tracking Solution

"If you're a developer, you've got to get TestTrack. Your bugs will hate you for it!" *Cool Tool of the Day*

Discover the tool today's top software developers are using to improve the quality of their **Macintosh** and **Windows** applications—**TestTrack.**

- Track bugs, feature requests, problems, customer information, and more.
- New! E-mail notifications—SMTP and MAPI.
- New! Import bugs via e-mail automatically.
- Produce concise reports.
- Multiple users, full security—link engineers, testers, managers, even tech writers.
- New! Improved help desk support.
- New! Deferred defect numbering, external attachment storage, customer bug histories, and much, much, more.
- Automatically route bugs to team members.
- Track the history of each bug.
- Save time and improve tech support by giving Solo Bug, TestTrack's stand-alone bug reporter, to your customers.

Only $169! 2+ for $149 each!
To order call 888-683-6456
or 513-683-6456

Download our demo today!
http://www.seapine.com

sales@seapine.com
http://www.seapine.com

Seapine Software

```
    *yourMove = move;
    gMoveNum = 2;
    return;
  }

  if (gMoveNum == 2) {  // *** MOVE 3
    if (opponentsMove.v < opponentsMove.h) {
      if (opponentsMove.v < -opponentsMove.h) {
  // top triangle
        move.h = 0;
        move.v = 4;
      } else {
  // right triangle
        move.h = -4;
        move.v = 0;
      }
    } else {
      if (opponentsMove.v >= -opponentsMove.h) {
  // bottom triangle
        move.h = 0;
        move.v = -4;
      } else {
  // left triangle
        move.h = 4;
        move.v = 0;
      }
    }
    pSq = &gBoard[TRANSLATE(move.h, move.v)];
    *pSq = _FRIEND;
    UPDATE_ENDPOINTS(pSq, gFirstStone, gLastStone);
    *yourMove = move;
    gMoveNum = 3;
    return;
  }

  gChangesEnd = gChanges;
  gCumCapturesEnemy +=
      MyFindCaptures(opponentCaptures, &gBoard[i],
                          _ENEMY, _FRIEND);
  for (pSq = gEstimatesStart; pSq != gEstimatesEnd; ++pSq)
    *pSq = 0;
  for (pSq = gFirstStone; pSq <= gLastStone; ++pSq)
    if (OCCUPIED(*pSq)) {
      d = gDirections;
      do {
        pNewSq = pSq + *d;
        if (EMPTY(*pNewSq)) {
          gEstimates[GET_INDEX(pNewSq)] += ESTIMATE_PLUS;
          pNewSq += *d;
          if (EMPTY(*pNewSq)) {
            gEstimates[GET_INDEX(pNewSq)] += ESTIMATE_PLUS;
          }
        }
      } while (++d != gDirectionsEnd);
    }

  for (pSq = gBoardStart; pSq != gBoardEnd; ++pSq)
    if (gEstimates[GET_INDEX(pSq)]) {
      i = GET_INDEX(pSq);
      gScore = gEstimates[i];
      gPreScore[i] = AddStone(-INFINITY, INFINITY, pSq,
                          _FRIEND, 0, gCumCapturesFriend,
                          gCumCapturesEnemy, gFirstStone,
                          gLastStone);
    }

  for (i=0; i<SEARCH_DEPTH; ++i)
    gKillers[i] = NULL;
  bestScore = -INFINITY;
  pBestMove = NULL;
  pSq = ChooseNextMove();
  while (pSq) {
    i = GET_INDEX(pSq);
    gScore = gEstimates[i];
    score = AddStone(-INFINITY, -bestScore, pSq, _FRIEND,
              gStartDepth, gCumCapturesFriend,
              gCumCapturesEnemy, gFirstStone, gLastStone);
    gEstimates[i] = -1; // searched
    if (score > bestScore) {
      bestScore = score;
      pBestMove = pSq;
    }
    pSq = ChooseNextMove();
  }

  if (bestScore == -INFINITY) {
    for (pSq = gBoardStart; pSq != gBoardEnd; ++pSq)
      if (EMPTY(*pSq) && !gEstimates[GET_INDEX(pSq)]) {
        gScore = 0;
        score = AddStone(-INFINITY, -bestScore, pSq, _FRIEND,
                    gStartDepth, gCumCapturesFriend,
                    gCumCapturesEnemy, gFirstStone, gLastStone);
        if (score > bestScore) {
          bestScore = score;
          pBestMove = pSq;
        }
      }
    if (bestScore == -INFINITY) {
      DebugStr("\p no move found - Pente");
      return;  //no move
    }
  }

  *pBestMove = _FRIEND;
  UPDATE_ENDPOINTS(pBestMove, gFirstStone, gLastStone);
  i = GET_INDEX(pBestMove);
  move.h = GET_X(i);
  move.v = GET_Y(i);
  *yourMove = move;
  ++gMoveNum;
 // find captures
  *numCaptures = MyFindCaptures(claimCaptures, pBestMove,
                      _FRIEND, _ENEMY);
  gCumCapturesFriend += *numCaptures;
  if (gCumCapturesFriend >= 5) {
    *claimVictory = true;
  } else {
    *claimVictory = MyFindFive(pBestMove, _FRIEND);
  }
}
```

TermPente

```
void TermPente(void) { }
```

ChooseNextMove

```
short * ChooseNextMove()
{
  short i, *pSq;
  short *found = NULL;
  short high = -INFINITY;
  for (pSq = gBoardStart; pSq != gBoardEnd; ++pSq) {
    i = GET_INDEX(pSq);
    if (gEstimates[i] > 0 && gPreScore[i] > high) {
      found = pSq;
      high = gPreScore[i];
    }
  }
  return found;
}
```

AddStone

```
// Alpha-beta search routine
// returns score of how good it is for color
// alpha and beta apply for the opponent after the move
// is made
// gScore is absolute: + for _FRIEND
short AddStone(short alpha, short beta, short *pSq,
        short color, short depth, short capturesFriend,
        short capturesEnemy, short *firstStone,
        short *lastStone)
{
  register short *pNewSq, *d;
  short x, bestScore, t, *pEnd;
  short open, *killer;
  short threats, blocks, vulnerable;
  short opponent = OPPONENT(color);
  short saveScore = gScore;

  START_SAVE;
  threats = blocks = vulnerable = 0;
  d = gDirections;
  do {
    pNewSq = pSq + *d;

    if (*pNewSq == color) {  // Next to friend
    if (d <= &gDirections[3]) {
      x = 1;
      open = 0;
      for (pNewSq += *d; *pNewSq == color; pNewSq += *d)
        ++x;
      if (EMPTY(*pNewSq))
        open = 1;
      for (pNewSq = pSq + *(d+4); *pNewSq == color;
            pNewSq += *(d+4))
        ++x;
      if (EMPTY(*pNewSq))
        ++open;
    } else {
      t = *(pSq + *(d-4));
      if (t == color)
        continue;
      x = 1;
      open = 0;
      if (EMPTY(t))
        open = 1;
      for (pNewSq += *d; *pNewSq == color; pNewSq += *d)
        ++x;
      if (EMPTY(*pNewSq))
        ++open;
    }

    switch (x) {
      case 1:  // 2-in-a-row
        gScore += CHAIN_SCORE2[color][open];
        if (open == 1)
          ++vulnerable;
        break;
      case 2:  // 3-in-a-row
        gScore += CHAIN_SCORE3[color][open];
        if (open == 2)
          threats += 2;          // tria = 2
        break;
      case 3:  // 4-in-a-row
        gScore += CHAIN_SCORE4[color][open];
        threats += open * 3;  // tessera = 6, half-open = 3
        break;
      default:  // 5-in-a-row (or more)
        gScore += CHAIN_SCORE5[color];
        threats = -99;            // game over
        break;
    }

  } else if (*pNewSq == opponent) {   // Next to enemy
    x = 1;
    for (pNewSq += *d; *pNewSq == opponent; pNewSq += *d)
      ++x;

    if (EMPTY(*pNewSq)) {
      blocks += BLOCK_SCORE[x];
    } else if (x == 2 && *pNewSq == color) {
      t = CAPTURE_SCORE;
      if (color != _FRIEND)
        t = -t;
      gScore += t;
      if (color == _FRIEND) {
        if (++capturesFriend >= 5) {
          threats = -99;  // game over
        }
      } else { // color == _ENEMY
        if (++capturesEnemy >= 5) {
          threats = -99;  // game over
        }
      }
      if (depth) {
        pNewSq = pSq + *d;
        PUSH_SQ(pNewSq);
        *pNewSq = _EMPTY;
        pNewSq += *d;
        PUSH_SQ(pNewSq);
        *pNewSq = _EMPTY;
      }
    }
  }
} while (++d != gDirectionsEnd);

if (threats < 0) {
// Game over
  bestScore = gScore;
```

StoneTable

You thought it was **just** a replacement for the List Manager ?

We lied, it is **much** more !

Tired of always adding just one more feature to your LDEF or table code ? What do you need in your table ?

Pictures and Icons and Checkboxes ?
adjustable columns or rows ?
Titles for columns or rows ?
In-line editing of cell text ?
More than 32K of data ?
Color and styles ?
Sorting ?
More ??

How much longer does the list need to be to make it worth $200 of your time ?
See just how long the list is for StoneTable.

Make StoneTable part of your toolbox today !
Only $200.00 MasterCard & Visa accepted.

StoneTablet Publishing
More Info & demo Voice/FAX (503) 287-3424
http://www.teleport.com/~stack stack@teleport.com

```
  if (color != _FRIEND)
    bestScore = -bestScore;
  goto RESTORE;
}

// Not forced to make a bad move
if (color == _FRIEND) {
  if (gScore < saveScore)
    gScore = saveScore;
} else if (gScore > saveScore)
  gScore = saveScore;
if (depth) {
// Add stone
  PUSH_SQ(pSq);
  *pSq = color;
  UPDATE_ENDPOINTS(pSq, firstStone, lastStone);
  --depth;
  gPreviousThreats = threats;
  bestScore = -INFINITY;
// Killer move?
  killer = gKillers[depth];
  if (killer && EMPTY(*killer)) {
    saveScore = gScore;
      bestScore = AddStone(-beta, -alpha, killer,
                      opponent, depth, capturesFriend,
                      capturesEnemy, firstStone, lastStone);
    gScore = saveScore;
    if (bestScore > alpha) {
      if (bestScore >= beta) {
        bestScore = -bestScore;
        goto RESTORE;
      }
      alpha = bestScore;
    }
  }

  pEnd = lastStone + gSE;
  if (pEnd >= gBoardEnd)
```

```
      pEnd = gBoardEnd - 1;
    pSq = firstStone - gSE;
    if (pSq < gBoardStart)
      pSq = gBoardStart;
    do {
      if (EMPTY(*pSq) && pSq != killer) {
        d = gDirections;
        do {
          pNewSq = pSq + *d;
          if (OCCUPIED(*pNewSq) &&
                (*(pNewSq + *d) == *pNewSq ||
                 *(pSq    - *d) == *pNewSq))
            break;
        } while (++d != gDirectionsEnd);
        if (d != gDirectionsEnd) {
          saveScore = gScore;
            t = AddStone(-beta, -alpha, pSq, opponent,
                  depth, capturesFriend, capturesEnemy,
                  firstStone, lastStone);
          gScore = saveScore;
          if (t > bestScore) {
            bestScore = t;
            if (t > alpha) {
              if (t >= beta) {
                gKillers[depth] = pSq;
                break;
              }
              if (depth >= gStartDepth-1)
                gKillers[depth] = pSq;
              alpha = t;
            }
          }
        }
      }
    } while (++pSq <= pEnd);
    if (bestScore == -INFINITY) {
      ++depth;
      goto TERMINAL;
    }
    bestScore = -bestScore;

  } else { // !depth
TERMINAL:
    bestScore = gScore;
    if (color != _FRIEND)
      bestScore = -bestScore;

  // Winning threats
    if (gPreviousThreats >= 5) {
      bestScore -= THREATS[gPreviousThreats];
    } else if (threats >= 5) {
      bestScore += THREATS[threats];
    } else if (gPreviousThreats == 4) {
      bestScore -= THREATS[gPreviousThreats];
    } else if (threats == 4) {
      bestScore += THREATS[threats];
    } else if (gPreviousThreats) {
      bestScore -= THREATS[gPreviousThreats];
    } else if (threats) {
      bestScore += THREATS[threats];
    }
    bestScore += blocks - vulnerable * VULNERABLE;
  }

RESTORE:
  while (POP) {
    pSq = (short *)TOP;
    *pSq = POP;
  }

  return bestScore;
}
```

MyFindCaptures

```
long MyFindCaptures(Capture capture[], short *pSq,
                     short us, short them)
{
  short i, *p1, *p2;
  short myCaptures = 0;
  short *d = gDirections;

  do {
    p1 = pSq + *d;
    if (*p1 == them) {
      p2 = p1 + *d;
      if (*p2 == them && *(p2 + *d) == us) {
        *p1 = *p2 = _EMPTY;
        i = GET_INDEX(p1);
        capture[myCaptures].stone1.h = GET_X(i);
        capture[myCaptures].stone1.v = GET_Y(i);
        i = GET_INDEX(p2);
        capture[myCaptures].stone2.h = GET_X(i);
        capture[myCaptures].stone2.v = GET_Y(i);
        ++myCaptures;
      }
    }
  } while (++d != gDirectionsEnd);
  return myCaptures;
}
```

MyFindFive

```
Boolean MyFindFive(short *pSq, short color)
{
  short x, *d, *pNewSq, i;

  d = gDirections;
  i = 0;
  do {
    x = 1;
    for (pNewSq = pSq + *d; *pNewSq == color; pNewSq += *d)
      ++x;
    for (pNewSq = pSq + *(d+4); *pNewSq == color;
          pNewSq += *(d+4))
      ++x;
    if (x >= 5)
      return true;
    ++d;
  } while (++i < 4);
  return false;
}
```

MT

Want to know what products are available for Mac OS development? Check out

Developer DEPOT™

<http://www.devdepot.com>

The Eddy Awards
Macworld Expo – San Francisco • January 1998

SPONSORED BY

For Macintosh Programmers & Developers

MacTech® MAGAZINE and Macworld

RAPID APPLICATION DEVELOPMENT TOOL

These products excel in helping developers create applications quickly.

Winner:

4th Dimension version 6

ACI US <http://www.acius.com>

Reason for award:

Lots of unbelievably positive user feedback. Strong cross platform RAD tool. Seamless cross-platform interfaces.

Description of product:

4th Dimension Version 6 (4D V6) is the Mac's most popular integrated database and Web server. Benefiting from built-in 4D capabilities, the combined database and Web server treat Web handling as a standard database task, enabling you to publish 4D databases on the Web quickly and easily.

Finalist:

Tools Plus 3.3

Waters Edge Software <http://www.interlog.com/~wateredg/>

Reason for award:

Good user feedback. Good learning tool. Good support. Good all around product.

Finalist:

Macintosh Common Lisp 4.1

Digitool, Inc. <http://www.digitool.com/>

Reason for award:

Strong and positive user feedback. Powerful and rich development tool using dynamic programming, multiprocessing, automatic resource allocation, and object oriented representation in a manner which is seamless and intuitive.

DEVELOPMENT ENVIRONMENT

These products are high quality development environments used by conventional core developers.

Winner:

CodeWarrior Professional, Release 2

Metrowerks, Inc. <http://www.metrowerks.com>

Reason for award:

Strong and positive user feedback. Excellent advances in last year on the product.

Description of product:

CodeWarrior is the only Integrated Development Environment (IDE) which enables you to edit, compile and debug C, C++, Java and Pascal programs for multiple target processors and operating systems. CodeWarrior's compilers produce fast, highly optimized code for Windows 95/NT running on x86, or Mac OS running on 68K or PowerPC processors. CodeWarrior's flexibility gives you more options.

Finalist:

FutureBASIC II

Staz Software, Inc. <http://www.stazsoftware.com/>

Reason for award:

Strong and positive user feedback. Professional quality product, yet good learning tool that conforms to established standards.

Finalist:

BBEdit 4.5.1

Bare Bones Software, Inc. <http://www.barebones.com/>

Reason for award:

Strong and positive user feedback. Powerful and memory-efficient application.

Tool for New Technologies

These products are important to developers on the bleeding edge — Rhapsody, Java, etc...

Winner:

CodeWarrior Latitude

Metrowerks, Inc. <http://www.metrowerks.com>

Reason for award:

Fills a critical gap in the Mac OS to Rhapsody transition. In my mind, it's as important as the 68k emulator was in the PowerPC transition. It allows developers to move code from Mac OS to native Yellow Box (i.e., Rhapsody) code at the pace they want to, not having to move everything at once.

Description of product:

CodeWarrior Latitude DR2

The road to Rhapsody starts with CodeWarrior Latitude. Don't be left behind when Rhapsody ships. CodeWarrior Latitude now supports the first developer's release of Rhapsody, so you can move your existing Mac OS 7.x or 8.x application to Rhapsody and UNIX today. CodeWarrior Latitude is a software library that enables software developers to create native Rhapsody and UNIX applications from existing Macintosh source code. Instead of running your application in the blue box, run it native.

Finalist:

Joy

AAA+ Software F&E GmbH. <http://www.aaa-plus.com>

Reason for award:

Joy Explorer is an excellent learning tool for Rhapsody — it allows you to look at how applications are built even if they aren't yours. Joy Developer has a rich heritage from OPENSTEP that is very useful in dynamically creating applications very quickly. Primary weakness is user interface which is very UNIX like.

Finalist:

Visual Café for Macintosh 1.0.2

Symantec Corporation <http://www.symantec.com>

Reason for award:

Rich set of visual tools for Java. Best Rapid Application Development type environment for Java on the Mac.

Developer Tool

These products should be part of nearly any developer's repetoir of tools.

Winner:

VOODOO 1.8

UNI SOFTWARE PLUS GMBH <http://www.unisoft.co.at>

Reason for award:

Strong and positive user feedback. Solves an important problem in a very elegant and powerful way. Remarkably stable product with solid integration with CodeWarrior.

Description of product:

VOODOO (Versions Of Outdated Documents Organized Orthogonally) is the ultimate version control solution you have been waiting for. It is a standalone version control tool with a neat graphical user interface offering many features for the simple and clear management of projects in which files evolve in numerous versions.

Finalist:

Installer VISE 4.6.1

MindVision, Inc. <http://www.mindvision.com>

Reason for award:

Good user feedback. Broad array of companies using the product. Very capable product of excellent quality.

Finalist:

Spotlight DR2

Onyx Technology, Inc. <http://www.onyx-tech.com>

Reason for award:

Strong and positive user feedback. Allows users to find bugs nearly immediately upon use. Needs interface work, but the debugger is very robust and useful.

List of 1997 Macworld Editors' Choice finalists Sponsored by Macworld

Hardware Product of the Year
Olympus D-600L, Olympus

Software Product of the Year
Mac OS 8, Apple Computer

Technology of the Year
PowerPC 750, IBM and Motorola

Cary Lu Technology Leadership Award
Stylus Photo, Epson

Best 3-D Graphics Program
Winner: ElectricImage Broadcast 2.7.5, Electric Image
Finalists: Infini-D 4.0, MetaCreations
Tree Professional 4.0, Onyx Computing

Best Art Application
Winner: Photoshop 4.0.1, Adobe Systems
Finalists: Illustrator 7.0.1, Adobe Systems
Painter 5, MetaCreations

Best Audio Software
Winner: FreeStyle 2.0, Mark of the Unicorn
Finalists: Peak 1.63, Bias
Cubase VST 3.5.2, Steinberg North America

Best Business Printer
Winner: magicolor 2 CX, QMS
Finalists: Phaser 560 , Tektronix
DocuPrint C55mp, Xerox

Best Business Productivity Tool
Winner: FileMaker Pro 4.0, Claris
Finalists: ClarisWorks Office, Claris
Wingz 2.1.1 Investment Intelligence Systems Group

Best Communications Hardware
Winner: QuickStream Pro, Sonic Systems
Finalists: YoYo Pro, Big Island Communications
InterJet, Whistle Communications

Best Consumer System
Winner: PowerCenter Pro 180, Power Computing
Finalists: StarMax 5000/300 , Motorola Computer Group
SuperMac C600x/280, Umax Computer

Best Data-Presentation Tool
Winner: Data Desk 6.0 Plus, Data Description
Finalists: ActiveStats 1.0, Benjamin/Cummings Science
Numbers & Charts 1.0.2, Adrenaline Software

Best Digital Camera
Winner: D-600L, Olympus America
Finalists: DC210 Zoom Digital Camera, Eastman Kodak
Digital Mavica, Sony Electronics

Best Digital-Video Hardware
Winner: MotoDV, Radius
Finalists: Optura, Canon USA
FireMax, ProMax

Best Digital-Video Utility
Winner: Media Cleaner Pro 2.0, Terran Interactive
Finalists: CineLook, DigiEffects
Studio Effects, MetaCreations

Best Display
Winner: MultiSync LCD2000, NEC Technologies
Finalists: Diamond Pro 91TXM, Mitsubishi
Multiscan 200sf, Sony Electronics
Optiquest V773, ViewSonic

Best Game
Winner: Myth: The Fallen Lords, Bungie Software
Finalists: Quake, MacSoft
Riven, Red Orb Entertainment

Best Gaming Hardware
Winner: F-16 Fighterstick, CH Products
Finalists: Pro Throttle, CH Products
Power3D, TechWorks

Best General-Purpose System
Winner: Power Macintosh G3/233 Desktop, Apple Computer
Finalists: PowerBook 2400, Apple Computer
PowerCenter Pro 210, Power Computing

Best Graphics Plug-In
Winner: Genuine Fractals, Altamira Group
Finalists: Eye Candy 3.0, Alien Skin Software
PhotoTools 2.0, Extensis

Best Input Device
Winner: UltraSlate, CalComp
Finalists: Orbit, Kensington Microware
PL-300, Wacom

Best Internet-Client Software
Winner: Emailer 2.0, Claris
Finalists: NetFinder 1.2.1, Peter Li & Vincent Tan
PGP for Personal Privacy Ver. 5.0, Pretty Good Privacy

Best Internet-Server Software

Winner: WebStar 2.1, Quarterdeck
Finalists: Lasso 2.0, Blue World Communications
Rumpus 1.1, Maxum Development

Best Multimedia-Authoring Software

Winner: Director 6 Multimedia Studio, Macromedia
Finalists: QuickTime VR Authoring Studio, Apple Computer
Authorware 4.0.1, Macromedia

Best Networking Software

Winner: CyberGauge 2.0, Neon Software
Finalists: PC MacLAN 6.2, Miramar Systems
SoftRouter 4.0, Vicom Technology

Best Personal Printer

Winner: Stylus Color 800, Epson America
Finalists: Color StyleWriter 6500, Apple Computer
Stylus Photo, Epson America

Best Professional System

Winner: SuperMac S900 Base with Umax Edition Maxpowr Pro+, Umax Computer
Finalists: Macintosh PowerBook G3, Apple Computer
PowerTower Pro 250, Power Computing

Best Projector

Winner: PowerLite 5000, Epson America
Finalists: LitePro 720, In Focus
Proscreen 4600 Endurance, Philips Electronics

Best Publishing Tool

Winner: QuarkXPress 4.0, Quark
Finalist: Acrobat Pro 3.01, Adobe Systems

Best Publishing Utility

Winner: Preflight Pro 1.0, Extensis
Finalists: Font Reserve 1.0, DiamondSoft
QX-Tools 2.0, Extensis

Best Scanner

Winner: Expression 636 Professional, Epson America
Finalists: PowerLook II, Umax Technologies
Astra 600S, Umax Technologies

Best Science/Engineering Tool

Winner: Mathematica 3.0.1, Wolfram Research
Finalists: MicroStation 95, Bentley Systems
MiniCAD 7.0.1, Diehl Graphsoft

Best Utility

Winner: Conflict Catcher 4.0.3, Casady & Greene
Finalists: Speed Doubler 8, Connectix
Virex with Speedscan 5.8, Dr Solomon's Software

Best Web Authoring Tool

Winner: CyberStudio 2.01, GoLive Systems
Finalists: Flash 2, Macromedia
PageSpinner 2.0.1, Optima System

World Class Awards

Impact Awards

One you know, the other you probably don't. But Apple interim CEO **Steve Jobs** and former Power Computing marketing manager **Mike Rosenfelt** both made a big difference in the Mac market.

Steve Jobs took Apple's reins in July and quickly made his mark on the company he co-founded. He reworked the board of directors, forged a controversial deal with Microsoft, and gave a new sense of direction to what had been a rudderless enterprise. But he also killed Apple's Mac OS licensing program, and by cutting out the clone vendors, he cut off a prime source of innovation in the Mac market. Whether you agree or disagree with his moves, he has clearly had an impact. While Macworld believes it is too soon to say whether Jobs will ultimately be good for Apple, we can't deny his impact on the Mac this past year.

Mike Rosenfelt never made the cover of »Time« or »Newsweek«, but the guerrilla marketing techniques he brought to Power Computing helped re-energize the Mac community in a way we haven't seen since a pirate flag flew above Apple's Cupertino headquarters many years ago.

He was the man behind the bullhorn at Power Computing's Macworld Expo pep rallies and the brains behind such memorable stunts as bungie jumping platforms in Boston and a fleet of Mac-promoting humvees in San Francisco. At a time when Apple seemed never to have its act together, Rosenfelt and Power Computing inspired us all to fight back for the Mac.

Lifetime Achievement Award

Richard Zulch is one of the unsung heroes in Macintosh history. He's the co-founder, chief technology officer, and vice president of engineering for Dantz Development, the company that gave us Retrospect. But his contributions to the Mac go far beyond back-up utility software.

Working as a pro bono consultant for Apple, Zulch played a key behind-the-scenes role in developing nearly every major storage driver for the Macintosh. You may take it for granted that your Mac works with hard drives from companies other than Apple; it was Richard Zulch who helped make it happen. Zulch was also a driving force behind HFS+, the new file system introduced with Mac OS 8.1, that lets you use your hard disk more efficiently. None of his work had a direct financial benefit for Dantz; instead, he gave freely of his time so that Macintosh users could have a better computing experience. His has truly been a lifetime of achievement on behalf of the Mac.

MT

by Jessica Courtney

STALKER SOFTWARE ANNOUNCES COMMUNIGATE PRO SERVER

After the successful launch of the Stalker Internet Mail Server, Stalker Software announced the CommuniGate Pro messaging server for the Rhapsody family of operating environments.

The new reincarnation of the CommuniGate integrated messaging system unifies the performance and standard-compliance of the SIMS with flexibility and powerful features of the modular MacOS CommuniGate System.

The CommuniGate Pro is based on the Rhapsody Foundation framework and can run under all operating systems supported with the Rhapsody "Yellow-box": Rhapsody for Power Macintosh, Rhapsody for Intel, and the Yellow Box for Windows NT.

The CommuniGate Pro design employs many modern OS techniques, especially multi-threading. Unlike other servers that use one separate thread to process a message, messages in the CommuniGate Pro are processed with 4-6 threads on their paths to destinations. For high-volume sites, such as major ISPs, this means a huge performance benefit, especially for multi-processor servers. The CommuniGate Pro server is completely SMP (symmetric multiprocessing) ready. The SMTP component of the CommuniGate Pro server incorporates all the anti-spamming features implemented in the CommuniGate SMTP and Stalker Internet Mail Server.
<http://www.stalker.com>.

LIGHTWORK DESIGN INTRODUCES NEW FAMILY OF LIGHTWORKS RENDERING AND UTILITY PLUG-INS FOR QUICKDRAW 3D

LightWork Design, the leading supplier of rendering engines for 3D design applications, is introducing its new family of LightWorks rendering and utility plug-ins for QuickDraw 3D at the MACWORLD Expo, Apple Developer Central, in San Francisco, January 6-9. The six new LightWorks plug-ins offer a wide range of features, including ray-tracing, hidden-line drawing, printing and QuickTime VR movie output.

The LightWorks plug-ins are available to QuickDraw 3D application developers for inclusion as part of their products. With the plug-ins, developers can offer their customers a range of new functionality without the time and expense of in-house development.

QuickDraw 3D application developers can choose from both rendering and utility plug-ins: All six LightWorks plug-ins for QuickDraw 3D are supported on Macintosh, Windows 95 and Windows NT systems.
<http://www.lightwork.com>.

MT

TCP/IP Scripting Addition ***The Internet Scripting Solution***

The **TCP/IP Scripting Addition** allows you to quickly develop Internet client/server applications using AppleScript®. If you want to script with MacTCP™ and Open Transport™, here's your solution!

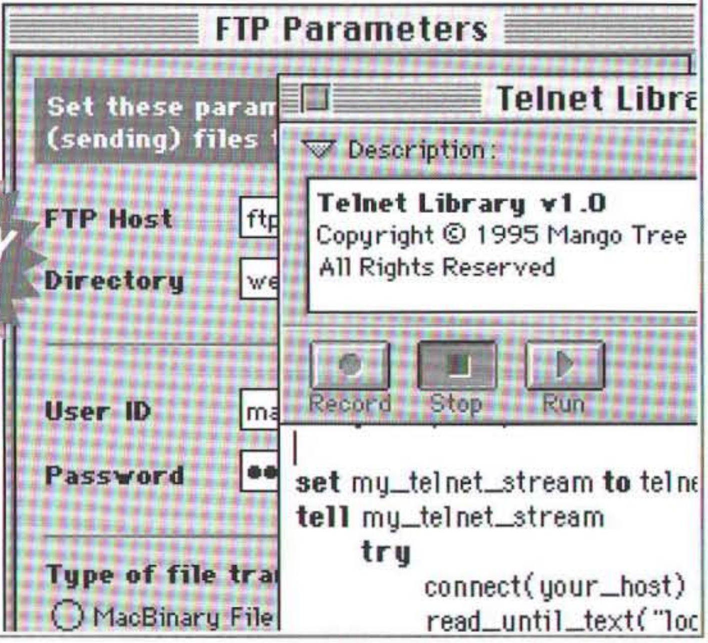
- Supports WebSTAR™, FaceSpan™, and HyperCard™
- The industry standard product for scripting Internet applications since 1993
- Sample scripts include FTP, HTTP, Telnet, Post Office, E–Mail and more
- Featured on the Apple® Internet Server

ONLY $49 SRP

Order now through Developer Depot at 800–MACDEV–1 or other mail order stores

Mango Tree Software, Inc.
Box 1057 • Brookline, Massachusetts USA 02146
Fax / Voice 617–327–8663
sales@mangotree.com • www.mangotree.com

All trademarks are properties of their respective holders.
Contact Mango Tree Software for site licensing and redistribution information.
Copyright © 1993–8 Mango Tree Software, Inc.



Telnet Library v1.0
Copyright © 1995 Mango Tree
All Rights Reserved

```
set my_telnet_stream to telne
tell my_telnet_stream
    try
        connect( your_host)
        read_until_text( "loc
```

The 13th Annual Conference for Leading Edge Developers

MacHack papers are the cornerstone of the premiere Macintosh technical conference. Papers cover a wide variety of cool topics. Anything from Adaptive Computing, the Business of Macintosh Development, Game Design, and Internet programming, to Rhapsody, Debugging, the Programming Subculture, Virtual Reality, and Mac OS internals and hacks.

Would you like to astound and amaze your peers and also get a free conference registration? If so, then share your in-depth knowledge of a technique, technology, philosophical point, or some other arcane knowledge and write a paper for **MacHack '98**!

Point your favorite email program to **machack-papers@machack.com** and send us your idea, abstract, questions, or other secrets. Abstracts are required by February 15, 1998.

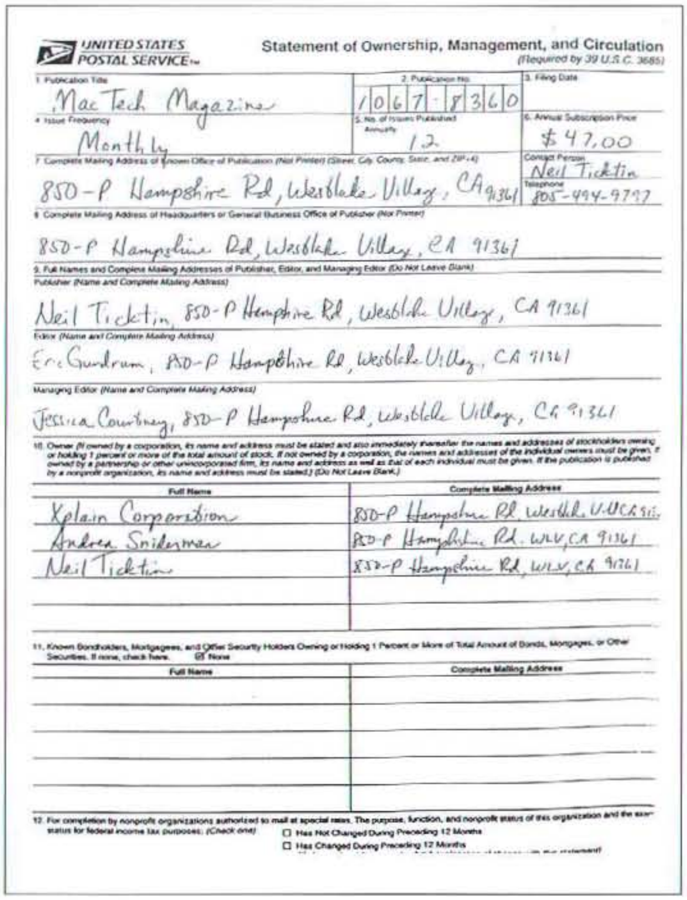
Attendance is limited — register early.

Bound proceedings & Best Hack CD will be distributed to all attendees.

For more information or to register:
Expotech, 1264 Bedford Rd., Grosse Pointe Park, MI 48230-1116
313-882-1824, **expotech@aol.com**, **http://www.machack.com/**

MacHack '98
June, 1998
Dearborn, Michigan

Macintosh is a trademark of Apple Computer. All others to their respective holders. MacHack is a trademark of Expotech, Inc. Not affiliated with the MacHax Group™.



UNITED STATES POSTAL SERVICE™

Statement of Ownership, Management, and Circulation
(Required by 39 U.S.C. 3685)

1. Publication Title: Mac Tech Magazine	2. Publication No.: 1067-8360	3. Filing Date:
4. Issue Frequency: Monthly	5. No. of Issues Published Annually: 12	6. Annual Subscription Price: $47.00
7. Complete Mailing Address of Known Office of Publication (Not Printer) (Street, City, County, State, and ZIP+4): 850-P Hampshire Rd, Westlake Village, CA 91361		Contact Person: Neil Ticktin; Telephone: 805-494-9797

8. Complete Mailing Address of Headquarters or General Business Office of Publisher (Not Printer): 850-P Hampshire Rd, Westlake Village, CA 91361

9. Full Names and Complete Mailing Addresses of Publisher, Editor, and Managing Editor (Do Not Leave Blank)

Publisher (Name and Complete Mailing Address): Neil Ticktin, 850-P Hampshire Rd, Westlake Village, CA 91361

Editor (Name and Complete Mailing Address): Eric Gundrum, 850-P Hampshire Rd, Westlake Village, CA 91361

Managing Editor (Name and Complete Mailing Address): Jessica Courtney, 850-P Hampshire Rd, Westlake Village, CA 91361

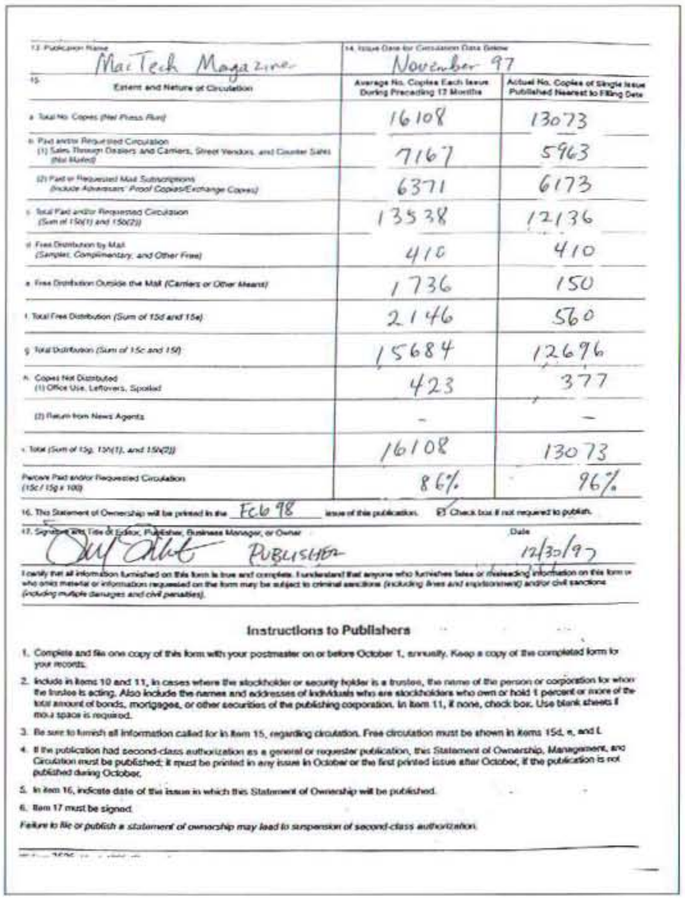
10. Owner (If owned by a corporation, its name and address must be stated and also immediately thereafter the names and addresses of stockholders owning or holding 1 percent or more of the total amount of stock. If not owned by a corporation, the names and addresses of the individual owners must be given. If owned by a partnership or other unincorporated firm, its name and address as well as that of each individual must be given. If the publication is published by a nonprofit organization, its name and address must be stated.) (Do Not Leave Blank.)

Full Name	Complete Mailing Address
Xplain Corporation	850-P Hampshire Rd, Westlake V. CA 913[illegible]
Andrea Sniderman	850-P Hampshire Rd. WLV, CA 91361
Neil Ticktin	850-P Hampshire Rd, WLV, CA 91361

11. Known Bondholders, Mortgagees, and Other Security Holders Owning or Holding 1 Percent or More of Total Amount of Bonds, Mortgages, or Other Securities. If none, check here. [✓ None]

Full Name	Complete Mailing Address

12. For completion by nonprofit organizations authorized to mail at special rates. The purpose, function, and nonprofit status of this organization and the exempt status for federal income tax purposes: (Check one)
☐ Has Not Changed During Preceding 12 Months
☐ Has Changed During Preceding 12 Months



13. Publication Name: MacTech Magazine	14. Issue Date for Circulation Data Below: November 97	
15. Extent and Nature of Circulation	Average No. Copies Each Issue During Preceding 12 Months	Actual No. Copies of Single Issue Published Nearest to Filing Date
a. Total No. Copies (Net Press Run)	16108	13073
b. Paid and/or Requested Circulation (1) Sales Through Dealers and Carriers, Street Vendors, and Counter Sales (Not Mailed)	7167	5963
(2) Paid or Requested Mail Subscriptions (Include Advertisers' Proof Copies/Exchange Copies)	6371	6173
c. Total Paid and/or Requested Circulation (Sum of 15b(1) and 15b(2))	13538	12136
d. Free Distribution by Mail (Samples, Complimentary, and Other Free)	410	410
e. Free Distribution Outside the Mail (Carriers or Other Means)	1736	150
f. Total Free Distribution (Sum of 15d and 15e)	2146	560
g. Total Distribution (Sum of 15c and 15f)	15684	12696
h. Copies Not Distributed (1) Office Use, Leftovers, Spoiled	423	377
(2) Return from News Agents	–	–
i. Total (Sum of 15g, 15h(1), and 15h(2))	16108	13073
Percent Paid and/or Requested Circulation (15c / 15g x 100)	86%	96%

16. This Statement of Ownership will be printed in the Feb 98 issue of this publication. ☑ Check box if not required to publish.

17. Signature and Title of Editor, Publisher, Business Manager, or Owner: [illegible] PUBLISHER — Date: 12/30/97

I certify that all information furnished on this form is true and complete. I understand that anyone who furnishes false or misleading information on this form or who omits material or information requested on the form may be subject to criminal sanctions (including fines and imprisonment) and/or civil sanctions (including multiple damages and civil penalties).

Instructions to Publishers

1. Complete and file one copy of this form with your postmaster on or before October 1, annually. Keep a copy of the completed form for your records.
2. Include in items 10 and 11, in cases where the stockholder or security holder is a trustee, the name of the person or corporation for whom the trustee is acting. Also include the names and addresses of individuals who are stockholders who own or hold 1 percent or more of the total amount of bonds, mortgages, or other securities of the publishing corporation. In item 11, if none, check box. Use blank sheets if more space is required.
3. Be sure to furnish all information called for in item 15, regarding circulation. Free circulation must be shown in items 15d, e, and f.
4. If the publication had second-class authorization as a general or requester publication, this Statement of Ownership, Management, and Circulation must be published; it must be printed in any issue in October or the first printed issue after October, if the publication is not published during October.
5. In item 16, indicate date of the issue in which this Statement of Ownership will be published.
6. Item 17 must be signed.

Failure to file or publish a statement of ownership may lead to suspension of second-class authorization.

MACTECH ONLINE

by Nicholas C. "nick.c" DeMello <online@mactech.com>

WHAT IS TCP/IP?

When Paul Baron first developed the concept that became today's Internet, he was trying to solve a military problem. His 1964 proposal described a communications network that had no central authority and could adapt to massive failures — therefore one that could not be easily disabled by enemy bombardment. Baron's solution was to break communications down into small pieces, each labeled with the address of a remote computer. These "packets" of information would then be passed through an intricate network of computers — like pails in a firefighter's bucket brigade. If a telephone line or fire hose is cut, the flow is interrupted, but if someone steps out of line in a bucket brigade or a computer goes offline, the system can pass it's cargo around the disruption. The more intricate the web of computer connections available in the network, the more reliable communication became. This packet based communications concept and the mechanisms for transferring packets along a network is the basis for the internet protocol — IP.

Vinton Cerf was a graduate student at UCLA in 1969, when he helped put the first computer on the internet. As more and more machines were added to the internet, standards needed to be developed for translating data from a system level into a format that could be transferred by the internet protocol. The InerNetworking Working Group (INWG) was founded in 1972 to develop these standards, with Vinton Cerf as the chairman. Two years later, Vint Cerf and Bob Kahn published "A Protocol for Packet Network Internetworking" which specified a standard protocol for converting a stream of data into a series of packets, for reassembling the packets into a stream of data, requesting lost packets be resent, and basically controlling communication through Baron's internet protocol. First titled the network control protocol (NCP), it was later renamed the transmission control protocol — TCP.

What kind of computer do these visionaries and founders of the information age use today? According to the MacAddict a 1996 television interview with Paul Baron showed him at his computer — a Power Mac 7100. During an interview for the Netinsider, Vint Cerf was asked what kind of computer he uses today. His response was an emphatic: "I'm a Mac Freak!"

A Short History of the Internet, by Bruce Sterling
<http://www.ubicom.com/history.html>
The Internet Valley, History of the Internet by Gregory R. Gromov
<http://www.internetvalley.com/intval.html>
Introduction to TCP/IP by H. Gilbert
<http://pclt.cis.yale.edu/pclt/comm/tcpip.htm>
RFC 1180: A TCP/IP Tutorial
<ftp://ds.internic.net/rfc/rfc1180.txt>
Paul Baron Appears on the Evangelist List of Famous Mac Users
<http://www.evangelist.macaddict.com/advocacy_action.html>
Interview with Vint Cerf, Internet Founder and "Mac Freak"
<http://www.netinsider.com/profile/vcerf/profile/>

USING TCP/IP ON A MAC

TCP/IP is accessed through Apple's Open Transport system. When you review the OT technology on Apple's web site (or in Inside Macintosh: Open Transport) you should realize that OT was designed to support transport independent code — your code use data streams to send information without knowing what kind of network the user is on (TCP/IP, AppleTalk, or other). Open Transport does allow you to design transport specific applications though, and especially with TCP/IP this may be desirable. The place to start for TCP/IP specific development is with Apple's OpenTransport/TCP dev note, which can be found on the same FTP site as the OT SDK. Also visit Mark Sproul's Open Transport Development web pages and Eric Scouten's Macintosh TCP/IP Programmer's Guide on the metrowerks web site.

John Norstad has made the Newswatcher source code publicly available. This extremely popular freeware USENET newsreader is compatible with MacTCP and OpenTransport, it's source is probably the single most valuable reference a Mac TCP/IP programmer can find. Look especially closely at the ftp.c, nntp.c, smtp.c, and net.c sources. These files are reusable, modular C code for implementing some of the most popular TCP/IP applications. Also check out Peter N Lewis' TCPExample code and his PNL TCP/IP libraries for implementing TCP/IP. This Pascal code is the foundation Peter's tremendously successful TCP/IP shareware products Anarchie and NetPresenz. Peter has developed Streamwatcher, a debugging tool that allows you to examine data in an Open Transport TCP stream. Once you have Streamwatcher, check out Mark Mentovai's Mac on the Net web site for other TCP/IP diagnostic utilities like Bryan Christianson's IP packet routetracer WhatRoute 1.4.3.

Apple's OpenTransport Website
<http://www.devworld.apple.com/dev/opentransport/>
Inside Macintosh, Open Transport and OT 1.2 Updates
<http://gemma.apple.com/techinfo/techdocs/mac/mac.html>
Open Transport/TCP Dev Notes & the OT SDK
<ftp://seeding.apple.com/ess/public/opentransport/>
The OpenTransport Mailing List, Hosted by Stairways Shareware
<http://www.stairways.com/mailinglists/opentpt.html>
Open Transport Development Pages by Mark Sproul
<http://msproul.rutgers.edu/macintosh/OpenTpt_Dev.html>
Macintosh TCP/IP Programmer's Guide, by Eric Scouten
<http://www.metrowerks.com/tcpip/>
John Norstad's Newswatcher Code
<ftp://ftp.acns.nwu.edu/pub/newswatcher/>
TCPExample & PNL TCP/IP Libraries by Peter N Lewis
<ftp://www.stairways.com/stairways/source/>
Mac TCP Watcher 2.0 and Streamwatcher 1.0 by Peter N Lewis
<http://www.stairways.com/mtcpw/>
<http://www.stairways.com/streamwatcher/>
Mac On The Net Archive (incl. WhatRoute and other TCP/IP Diagnostic Utils)
<http://www.moxienet.com/macnet/>

These and other links are archived on the MacTech Online web pages at <http://www.mactech.com/online/>. MT

TIPS & TIDBITS

by Steve Sisak

Logical Names

Unfortunately, the Macintosh doesn't have a system-level mechanism for dealing with the notion of logical paths. Aliases don't work when files get moved, and using paths with device names doesn't work because everybody uses different device names (besides, device names aren't guaranteed to be unique).

As a kluge for MPW Shell, AppleScript and Macintosh Common Lisp, we create a folder in the system folder called "Logical Names" and place aliases to other files and folders in this folder. The name of each alias is then defined, in some application specific way (q.v, below), as a symbol bound to the file found by resolving the alias. This allows us to put aliases like "ProjFolder" or "MoreFilesFolder" in the logical names folder and have a chance of being able to use build scripts, Makefiles, etc. on other folks' machines without having to do a major re-write of the files in question.

Usage

To make use of any of the specific implementations described below, you need first create your logical names folder. This folder must be immediately in your system folder, and it must be named "Logical Names" (note the space).

Then, create some useful aliases in this folder. For example, we typically place the following in our logical names folders:

```
TempFolder - an alias to the RAM Disk
HomeFolder - an alias to the folder containing user folders
ProjFolder - an alias to the folder containing project folders
ArchivesFolder - an alias to the folder containing archives
TransferFolder - an alias to a folder used as transient storage
```

You may place aliases within folders within the logical names folder, though you should avoid name conflicts between folders.

MPW Shell

This implementation is distributed in the file: UserStartup•!LogicalNames

For the MPW Shell implementation of Logical Names, we use a start-up script to create an environment variable for each alias in the logical names folder. The name of each variable is the same as the name of the alias. The value of each variable is the resolution of the alias. In addition, the LogicalNames variable is set to a list of the installed logical names.

To make use of this script, simply place it in your MPW folder (leaving it named "UserStartup•!LogicalNames").

For example, if you've created the "ArchivesFolder" logical name alias, once you launch MPW the following should report the full path to your archives folder:

```
echo {ArchivesFolder}
```

AppleScript

This implementation is distributed in the file: LogicalNames.as.

For the AppleScript implementation of LogicalNames, we use a function to lookup and resolve a logical name alias given its name. For this implementation only, if the alias is in a sub-folder of the logical names folder, then a partial path must be supplied. LogicalNames.as also supplies a utility function which converts an item into a full path string.

To make use of these functions, you must first load and compile LogicalNames.as using the Script Editor. Then, you must save the compiled script as a Compiled Script to the file "LogicalNames" in your Scripting Additions folder. Now it's in a well-known location. Then, insert the following at the beginning of every script in which you want to use logical names:

```
tell application "Finder"
   set LogicalNames to ¬
      (load script file ¬
         (name of startup disk & ":" & ¬
            name of system folder & ":" & ¬
            "Extensions:Scripting Additions:LogicalNames"))
end tell
```

This will bind the LogicalNames variable to the (previously saved) compilation of the LogicalNames.as script, from which you may use the logicalNameGetItem(aName) and fullPathFromItem(anItem) functions.

For example, if you've created the "ArchivesFolder" logical name alias, a script such as the following should report the full path to your archives folder:

```
tell application "Finder"
   set LogicalNames to ¬
      (load script file ¬
         (name of startup disk & ":" & ¬
            name of system folder & ":" & ¬
            "Extensions:Scripting Additions:LogicalNames"))
end tell
set myArc to logicalNameGetItem("ArchivesFolder") of LogicalNames
fullPathFromItem(myArc) of LogicalNames
```

Macintosh Common Lisp

This implementation is distributed in the file: logical-names.lisp

MCL has a concept similar to logical names in its "logical hosts", but no reasonable built-in mechanisms for defining them in terms of the generalized Mac file system. So, for the MCL implementation of Logical Names, we simply provide functions for defining a logical host for each alias in the logical names folder. The name of each logical host is the same as the name of the alias. The target of each logical host is the resolution of the alias.

To make use of these functions, simply place logical-names.lisp in your MCL folder, then add the following to your init.lisp (or .fasl/.pfsl) file:

```
(load "logical-names.lisp")
(define-all-logical-names)
```

For example, if you've created the "ArchivesFolder" logical name alias, once you launch MCL the following should report the full path to your archives folder:

```
(full-pathname (make-pathname :host "ArchivesFolder"))
```

Mike Webb <mjw@codewell.com> and
Jeff Mallatt <jjm@codewell.com>

SERVICES

The Law Office of Bradley M. Sniderman

California Lawyer focusing on Intellectual Property, Corporate, Commercial and Contract law, as well as Wills and Trusts.

If you are looking to protect your software with Copyright or Trademark protection, or if you need help establishing or maintaining your business, please give me a call or an e-mail. Reasonable fees.

(310) 553-4054

Brad@Sniderman.com

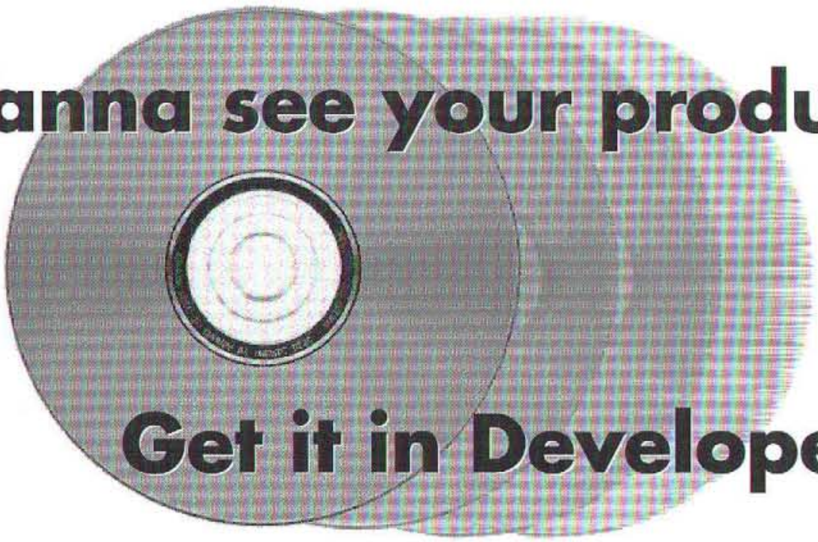
Visit MacTech® Magazine's Web site!

http://www.mactech.com

http://www.scientific.com

Professional software developers looking for career opportunities should check out our web site. We offer nationwide employment assistance, resume help, marketability assessment, and never a fee to the applicant. 800-231-5920, Fax 800-757-9003 eMail:das@scientific.com

Scientific Placement, Inc.



Wanna see your product *move?*

Get it in Developer Depot!

The fastest, cost effective way to get product off your shelves.

For information: • Voice: 805/494-9797 • Fax: 805/494-9798 • E-mail: marketing@devdepot.com

ADVERTISER/ PRODUCT LIST

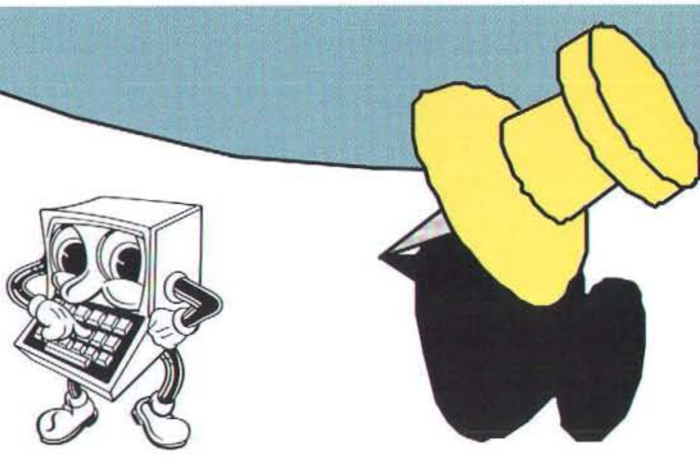
LIST OF ADVERTISERS

Aladdin Knowledge Systems Ltd.	5
Aladdin Systems, Inc.	7
APS Technologies	21
Bare Bones Software, Inc.	IBC
Bowers Development	43
Creative Soloutions	37
Developer Depot	65
Digital Technologies International	1
Genieworks	41
Mac Hack	60
MacTech CD ROM	25
MacTech Web Site	32
Main Event	29
Mango Tree	59
Mathemaesthetics, Inc.	11
Metrowerks	BC
Micro Macro Technologies Ltd.	9
Nombas, Inc.	15
Onyx Technology, Inc.	49
OpenBase International	13
PrimeTime Freeware	27
Quadrivio Corporation	23
Seapine Software, Inc.	51
Scientific Placement	63
SNAP	47
Snowbound Software Corporation	28
StoneTablet Publishing	53
Symantec	IFC
Tip Top	33
Water's Edge Software	50

LIST OF PRODUCTS

AppMaker • Bowers Development	43
BBEdit • Bare Bones Software, Inc.	IBC
Classified • Scientific Placement, Inc.	63
Classified • Sniderman, Bradley M.	63
CodeWarrior™ • Metrowerks	BC
Developer Tools • APS Technologies	21
Developer Tools • Developer Depot	65
FaceSpan • Digital Technologies Int.	1
General Edit • Quadrivio Corp.	23
InstallerMaker • Aladdin Systems, Inc.	7
MacHASP • Aladdin Knowledge Systems Ltd.	5
MicroGuard Plus™ • Micro Macro Technologies, Ltd.	9
MK Linux/MacPerl • PrimeTime Freeware	27
Objective Everything • Tip Top	33
OpenBase • OpenBase International	13
PrimeBase • SNAP	47
RasterMaster 6.0 • Snowbound Software Corporation	28
Resourcerer® 1.2 • Mathemaesthetics, Inc.	11
ScriptEase : Intergration SDK • Nombas	15
Scripter • Main Event	29
Serial/Parallel Expansion Cards • Creative Soloutions	37
Spot Check • Genieworks	41
Spotlight™ • Onyx Technology, Inc.	49
StoneTable • StoneTablet Publishing	53
TCP/IP Scripting Addition • Mango Tree	59
TestTrack™ • Seapine Software, Inc.	51
Tools Plus™ • Water's Edge Software	50
Visual Cafe™ • Symantec	IFC

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.



Developer To Do List:

- [✓] Prioritize project list & generate schedule
- ☐ Pick up new MacTech CD-ROM v. 1-12
- ☐ Check out all the Dev Depot specials
- ☐ Tell the gang about Dev Depot deals
- ☐ Stock up on the latest books & utilities
- ☐ Enter Dev Depot info into contact file

Developer Depot
PO Box 5200
Westlake Village, CA 91359-5200

Voice: 800/MACDEV-1
Outside US/Canada: 805/494-9797
Fax: 805/494-9798
E-mail: orders@devdepot.com
http://www.devdepot.com

Developer DEPOT™

develop

by Éric Simenel

And then SOM…

IBM System Object Model (SOM), which was introduced on Mac OS with OpenDoc, has become even more important to developers with the introduction of the Contextual Menus in Mac OS 8. It's not restricted to the development of Contextual Menus plug-ins, though. SOM is also available to all developers and solves a lot of common issues which plague real-life project development, management and maintenance. This article will show how SOM can ease your product development and save your precious time.

Since most developers develop in C++ or other object-oriented languages these days, most of you know the obvious reasons why object-oriented programming is better than the previous models. The main strengths (which are also its definitions) of OOP are encapsulation, polymorphism, and inheritance, all of which are conducive to easy reuse.

That's theory. In real life, we deal with different development environments which, even when they're dealing with the same language, are not always compatible because of implementation choices, and are usually even less compatible when dealing with different languages. Even when staying with the same development environment, it changes over time, and it's always trouble to reintegrate old code in new projects, and sometimes, it's even trouble trying to reopen an old project for maintenance with a current development environment.

So, a usual project these days involves many developers who have different tastes in development environments and languages, old legacy source codes written in different languages under different development environments, and sometimes old legacy binary codes whose source codes are no longer available.

Although SOM is not the universal cure to these problems, it does bring a lot of relief, and even if you're a single developer using only one language, SOM can help you manage your project by breaking it into small easily reusable pieces.

GETTING SOM

SOM provides developers with the advantages of both object-oriented programming and shared libraries. The main advantage is that whether you're just using a SOM class from a library or inheriting from it, if you later replace this library with a more recent version, you don't have to recompile or rebuild all the client code. When you stop and think about it, this feat is a real breakthrough from what we've been dealing with until now. The second advantage is SOM's language independence, which allows developers to use their favorite environment and still mesh with other people's code. Due to its rather recent emergence in the Mac OS, the preferred (and only) language used with SOM is C++, but it may change in the future.

This article will cover the basics of SOM encapsulation, polymorphism, and inheritance, the use of a SOM library in an application, and in another SOM library, multiple aspects of versioning, dealing with exceptions, etcetera. This article is aimed at giving you a kick start, covering the basics aspects of SOM and how to use it for fun and profit; it will explain in some places the internal works, in case you're interested, but not everywhere. If you're interested in more knowledge about SOM and its internals, then you should read the manuals provided where SOMObjects™ for Mac OS is distributed (for example, on the Mac OS SDK CD). This knowledge is not an absolute need, mostly you can just program by example (and a lot of examples

Éric Simenel is really happy he transferred to Cupertino's DTS from Paris' DTS. Aside from the fact that he received a great welcome from his current colleagues, he's getting much more sun here than there. And, due to his constant location here, he has easier access to comic books conventions, so he has completed many runs: the current mark is at 23,000 and counting.

are provided in this article). An alternate title for this article could have been "SOM for C++ developers...". When I refer in this article to the Users Guide, it means the Users Guide found in the documentation folder of SOMObjects.

Throughout this article, I'll use the following example to illustrate the different techniques:

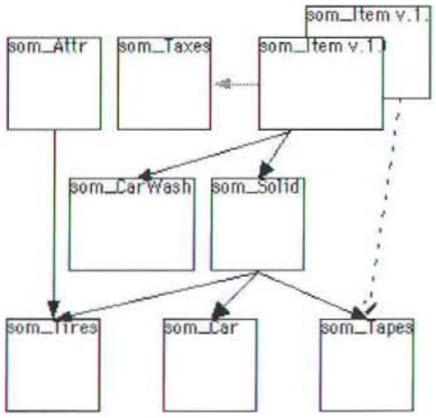


Figure 1. *SOM Classes of the example.*

Where som_Taxes (***encapsulation***) is a SOM class used by som_Item (***usage of a SOM lib within a SOM lib***). som_Solid (***inheritance***) and som_CarWash (***Meta Classes***) both inherit from som_Item. som_Car (***exception handling***) inherits from som_Solid, som_Tires (***multiple inheritance***) multiple inherits from som_Solid and som_Attr, and som_Tapes (***versionning***) inherits from som_Solid but does interesting things when som_Item v1.1 is present.

Each class is there to illustrate one interesting point at a time (to prevent confusion of the issues). To understand the example, you should note that I have assumed that we're located in a country where products are submitted to sales tax but services are not.

I've been using the Direct-To-SOM capabilities of MetroWerks CodeWarrior MW C/C++ (CodeWarrior Pro release). I verified that MPW MrCpp also has the same capabilities, but I don't cover its usage in this article. I also verified that you can build 68K SOM libraries, but do not cover that in this article either as there are only minor differences in the project settings.

Throughout this article, all the listings have been purged of irrelevant (for this article) lines, such as debugging information, to hilite the more interesting parts. Look at the real code provided with this article to see the complete sources.

Encapsulation

Let's say that you have old legacy source or binary code. Each time you want to use it in a new project it may be a pain to integrate, because things change over time. If it's binary code, it may be 68K code, for which you have to construct UniversalProcPtrs. If it's source code, it may be in a different language than the one you're currently using. Currently, if that code is callable from C/C++, then you can encapsulate it in 1 or more SOM classes distributed in 1 or more SOM shared libraries. In my example, that would be the som_Taxes SOM class.

The main advantage of encapsulation in a SOM wrapper is to enable you to use this old legacy code in all your new projects, with a nice interface. And, eventually, if you decide to rewrite all or part of it, all the projects which have been using this code won't have to be rebuilt to continue to work. The SOM wrapper really isolates the interface from the implementation. Whereas, if you were to keep this code the way it is, with maybe just a nice new set of headers to be able to use it easily in new projects, and if you decide later to modify all or part of it, then you would have to rebuild everything.

The costs involved are not only a development issue but also a sales issue, in the first case, since we're dealing with shared libraries, you just have to ship your customers the new version of the SOM library containing the old legacy code, now updated. In the second case, after having rebuild everything, you have to ship everything.

Creating a SOM library is pretty straightforward. Without the Direct-To-SOM capabilities of the most recent C++ compilers, we would have had to write first an .idl file, then go through MPW somipc to generate the .xh, .xih and .cpp files, and then modify the .cpp file according to what had to be done. With the Direct-To-SOM capabilities, it's much quicker: we simply write a .hh header file (.hh is just a style convention to differentiate them from simple .h header files, but there are no other differences) and the corresponding .cpp file. As a reminder, let's take a look at the .idl we would have written for straight SOM:

Listing 1
som_Taxes.idl

```
#include <somobj.idl>

module CalcTaxes
{
    interface som_Taxes : SOMObject
    {
    long CalcTheTax(in long value, in short kind);
#ifdef __SOMIDL__
        implementation
        {
            majorversion = 1;
            minorversion = 0;
            functionprefix = som_Taxes__;
            override: somInit;
            releaseorder: CalcTheTax;
            long tax;
        };
#endif //__SOMIDL__
    };
};
```

Compare this with the .hh we are writing for Direct-To-SOM compilation:

Listing 2
d2som_Taxes.hh

```
#include <somobj.hh>

class CalcTaxes_som_Taxes : public virtual SOMObject {
public:
                    CalcTaxes_som_Taxes();
    virtual         ~CalcTaxes_som_Taxes();
    virtual long CalcTheTax(INOUT Environment *ev, IN
long value, IN short kind);
private:
    long fTax;
#if __SOM_ENABLED__
    #pragma SOMReleaseOrder (CalcTheTax)
    #pragma SOMClassVersion (CalcTaxes_som_Taxes, 1, 0)
    #pragma SOMCallStyle IDL
#endif
};
```

Comparing both .idl and .hh, the meaning of the **SOMReleaseOrder** and **SOMClassVersion** pragmas are pretty much obvious. The **SOMCallStyle** pragma can take 2 arguments, IDL and OIDL. If we want to use the exceptions mechanism (more on that later in this article), it is imperative we use the IDL argument, so it's a good idea to get into this habit. This means that each method must have **Environment *ev** as its first parameter.

The first difference between a SOM class and a straight C++ class is that the constructor (ie. **CalcTaxes_som_Taxes()** in this example) for a SOM class can't have any arguments (we'll see later in this articles how to use an equivalent mechanism with Meta Classes). We can also override somInit, which is a method defined in SOMObject, but it can't take any arguments either. And anyway, you can't use an efficient exceptions mechanism with either constructor or somInit, so we'll see in the next example how you should use Initialize and Uninitialize methods to properly set up and unset your objects. In the case of som_Taxes, since it doesn't do much, we simply use a constructor which can't fail.

In this example, there's only one new method, **CalcTheTax**. **CalcTaxes_som_Taxes** is inheriting from **SOMObject** which is the root for SOM classes, and has methods like **somGetClass** and others which we'll see uses for later. The keyword **IN** found in the parameter list of **CalcTheTax** means that the parameter is passed to the method and, even if modified, not returned. As we'll see further below, the keywords **OUT** and **INOUT** can also be used, **OUT** meaning that the parameter doesn't receive an initial value and is returned by the method, **INOUT** meaning that the parameter is passed to the method with an initial value, and that the method can return another value. In fact, **IN** , **OUT** and **INOUT** are just for reading purposes, since these macros expand to nothing, they're just here as a reminder for SOM's **in**, **out** and **inout** keywords. But the same rules don't apply. A SOM **out long value** would be transformed in the .cpp as **long *value**. Here we have to say **OUT long *value**. In addition, the used SOM classes won't get an extra * (see the note on page 61 of develop 26).

SOM Objects and SOM Classes

SOM is a dynamic environment where classes themselves are instantiated as objects in memory. Those special objects are called class objects to differentiate them from simple objects, but there are objects nonetheless. Notwithstanding its name, **SOMObject** is a SOM class. **SOMClass** is also a SOM Class inheriting from **SOMObject**. To simplify complex matters, let's just say that when the SOM Runtime starts, it instantiates a class object of the class **SOMClassMgr**, a class object of the class **SOMObject**, and a class object of the class **SOMClass**. When you are instantiating an object of the class **CalcTaxes_som_Taxes** for the first time, SOM actually instantiates a class object for your class, and then, the object you requested. The class object of your class is instantiated only once, whatever the number of objects of this class you are instantiating. If you are interested in more details, please refer to chapter 2.1 of the Users Guide. Throughout this article, I made an effort to distinguish a SOM object from its SOM class, where it made sense, but mostly they're the same. When I'm writing about a method, for example, I may refer to it as the "SOM object method" or "SOM class method", but it doesn't matter much.

The next step is to write the .cpp file. It's in that .cpp file that you are going to either include or link to your old legacy code:

Listing 3
d2som_Texes.cpp

```
CalcTaxes_som_Taxes::CalcTaxes_som_Taxes()
    {
    fTax = 8;
    }

long CalcTaxes_som_Taxes::CalcTheTax(INOUT Environment
*ev, IN long value, IN short kind)
    {
    if (kind == 0) return value;
    else return(value + ((value * fTax) / 100));
    }
```

A big difference between C++ and SOM is that, by definition, all fields of a SOM class are private to that object, and all methods declared in the releaseorder list must be public and virtual. That means that if you want classes, inheriting from your class, accessing your fields, you have to provide accessors for them.

The next step is simply to build the SOM shared library. Using MetroWerks CodeWarrior, we'll need an extra file which contains the only symbol which has to be exported, for CFM (Code Fragment Manager) and SOM to be happy. This symbol is the complete name of your SOM class concatenated at the end with ClassData, ie. **CalcTaxes_som_TaxesClassData**. The name of the file must end with .exp and may be put in the project, ie. d2som_Taxes.exp. If there are more than one SOM class defined in a particular library, then all the **ClassData** symbols should be listed in the .exp file. Another way to achieve symbol export is to use the **#pragma export** directive in your .hh file as in the following example (see the Solid project for more details):

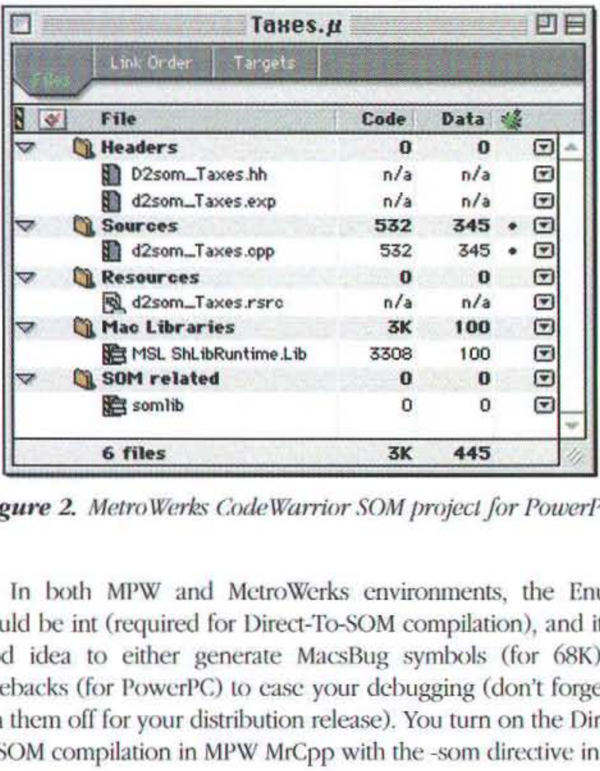
```
#pragma export on
class MSolid_som_Solid;
#pragma export off
```

Since the #pragma direct_to_som directive is on, the development environment does the right thing and adds the ClassData extension automatically.

You then create a new project based on the "ANSI C++ Console PPC (DLL)" (Pro 1) or "Std C++ Console PPC (DLL)" (Pro 2) stationery. You add the .cpp source file, the .exp file, and for convenience, the .hh file (duplicate the .h Target preference, and change the extension to .hh) and other source or resource files if you need. You also add the somlib shared library which contains the SOM code, and you modify the following settings:

- PPC Target: you select Shared Library as project type, you type in the name you chose, and you type cfmg as creator, unless you're providing your own bundle.
- PPC Linker: you clear all Entry Points fields
- PPC PEF: you select either the "use the ".exp" file" or "use #pragma" item in the Export Symbols popup menu depending on the way you desire to export symbols.

Depending on the code you either include or link to, you may or may not get rid of a lot of unuseful libraries automatically put in the project stationery. In my case, I only need to keep MSL ShLibRuntime.Lib and you build the library... (1421 bytes).



File	Code	Data
Headers	0	0
D2som_Taxes.hh	n/a	n/a
d2som_Taxes.exp	n/a	n/a
Sources	532	345 •
d2som_Taxes.cpp	532	345 •
Resources	0	0
d2som_Taxes.rsrc	n/a	n/a
Mac Libraries	3K	100
MSL ShLibRuntime.Lib	3308	100
SOM related	0	0
somlib	0	0
6 files	3K	445

Figure 2. *MetroWerks CodeWarrior SOM project for PowerPC.*

In both MPW and MetroWerks environments, the Enums should be int (required for Direct-To-SOM compilation), and it's a good idea to either generate MacsBug symbols (for 68K) or tracebacks (for PowerPC) to ease your debugging (don't forget to turn them off for your distribution release). You turn on the Direct-To-SOM compilation in MPW MrCpp with the -som directive in the command line, and you can turn it on (popup menu) in the C/C++ Language preference panel in MetroWerks CodeWarrior. You can alternatively use the #pragma direct_to_som on, if you prefer.

USING THE SOM SHARED LIBRARY

It's very simple to use the SOM shared library in your application project (or another SOM shared library project, in this case it's GeneralItem_som_Item), you just have to add the TaxesSOMLib to your project window, add the #include "d2som_Taxes.hh" directive in your .cpp source file, allocate the SOM object with a new, deallocated it with a delete, and use it as if it were a C++ object everywhere you need it.

But let's take a closer look at the d2som_Item project first. The .hh is:

Listing 4
d2som_Item.hh

```
#include "D2som_Taxes.hh"

class GeneralItem_som_Item : public virtual SOMObject {
public:
                    GeneralItem_som_Item();
    virtual        ~GeneralItem_som_Item();
    virtual void Initialize(INOUT Environment *ev);
    virtual void Uninitialize(INOUT Environment *ev);
    virtual void SetTheBTPrice(INOUT Environment *ev, IN
long thePriceBeforeTax);
    virtual void SetProductOrService(INOUT Environment
*ev, IN short kind);
    virtual long CalcTheATPrice(INOUT Environment *ev);
private:
    short fProductOrService;
    long  fBeforeTaxValue;
    CalcTaxes_som_Taxes*  fTheTax;
#if __SOM_ENABLED__
    #pragma SOMReleaseOrder (CalcTheATPrice,
SetTheBTPrice, SetProductOrService,
                                   Initialize, Uninitialize)
    #pragma SOMClassVersion (GeneralItem_som_Item, 1, 0)
    #pragma SOMCallStyle IDL
#endif
};
```

Note that you can't call anything which might throw an exception from somInit or somUninit or from the constructor or destructor of the C++ class. That's why it's usually a good idea to add to all your SOM classes the methods Initialize and Uninitialize which will be able to throw exceptions and which will be called just after the allocation and just before the deallocation of the SOM object.

In the example below, we do create the CalcTaxes_som_Taxes object in the Initialize method. The interesting part of d2som_Item.cpp is:

Listing 5
d2som_Item.cpp (extract)

```
void GeneralItem_som_Item::Initialize(INOUT Environment *ev)
     {
     fTheTax = new CalcTaxes_som_Taxes;
     }

void GeneralItem_som_Item::Uninitialize(INOUT Environment *ev)
     {
     delete fTheTax;
     }

long GeneralItem_som_Item::CalcTheATPrice(INOUT Environment *ev)
     {
     return(fTheTax->CalcTheTax(ev, fBeforeTaxValue,
fProductOrService));
     }
```

After that, we set up our new project file as before, without forgetting to add the TaxesSOMLib, and we do a build Now we can test our new SOM shared library.

To test all the SOM shared libraries in this article, I built one testing application, TestLibraries, which uses console output to hilite only the SOM coding. Its project file contains, of course, somlib and all the SOM shared libraries directly used by the application. For instance, it uses directly the som_Item object, but not the som_Taxes object which is used indirectly via the som_Item object, so only the ItemSOMLib is included in the project window and not the TaxesSOMLib. Of course, when the application is launched, the Process Manager has to find the TaxesSOMLib somewhere (or else you will get the usual Finder message saying that the TaxesSOMLib couldn't be found, and the application won't launch). So for pure ease of development, I set up projects to have all my shared libraries (some through aliases) and the application in the same folder. If you're interested in knowing of others places the shared libraries can go and still be found by the Process Manager, look at the "Mac OS Runtime Architectures" document available on the "Reference Library" Developer CD.

The interesting part of the source code of TestLibraries for the manipulation of GeneralItem_som_Item is:

Listing 6
TestLibraries.cpp (extract)

```
#include "d2som_Item.hh"

main()
     {
     long thePrice;
     AutoInitEnvironment sev;
     Environment *ev = (Environment *)&sev;
     GeneralItem_som_Item* theItem = new
GeneralItem_som_Item;
     theItem->Initialize(ev);
     theItem->SetTheBTPrice(ev, 100);
     theItem->SetProductOrService(ev, 0);
     thePrice = theItem->CalcTheATPrice(ev);
     printf("General Item, service, thePrice = %ld\n",
        thePrice);
     theItem->SetProductOrService(ev, 1);
     thePrice = theItem->CalcTheATPrice(ev);
     printf("General Item, product, thePrice = %ld\n\n",
        thePrice);
     theItem->Uninitialize(ev);
     delete theItem;
     }
```

And the execution printout is:

```
General Item, service, thePrice = 100
General Item, product, thePrice = 108
```

Since we need an Environment* variable to pass to the SOM methods, we simply use the specially-defined-for-Direct-To-SOM-C++-compilation AutoInitEnvironment structure. We could also use Environment* ev = somGetGlobalEnvironment();. When you are writing an application, you care only about one Environment structure, so you might as well be using the one provided by the somlib shared library (somGetGlobalEnvironment). If you are writing threads, then it is imperative to allocate one Environment variable on the stack of each thread (AutoInitEnvironment) or else you will be in trouble.

INHERITANCE AND POLYMORPHISM

Now that we have defined our som_Item base class, let's inherit from it.

All the previous SOM classes inherited from SOMObject, but this time we'll inherit from GeneralItem_som_Item, leading to the following declaration, implementation and usage:

Listing 7
MSolid_som_Solid declaration, *implementation and usage*

```
class MSolid_som_Solid  : public virtual
GeneralItem_som_Item {
public:
     virtual    void     SetQuantity(INOUT Environment *ev,
IN short howmany);
     virtual    void     SetUnitPrice(INOUT Environment
*ev, IN short howmuch);
// overriding
     virtual    void     Initialize(INOUT Environment *ev);
     virtual    long    CalcTheATPrice(INOUT Environment *ev);
private:
     short     fQuantity;
     short     fUnitPrice;
#if __SOM_ENABLED__
     #pragma SOMReleaseOrder (SetQuantity, SetUnitPrice)
     #pragma SOMClassVersion (MSolid_som_Solid, 1, 0)
     #pragma SOMCallStyle IDL
#endif
};

void MSolid_som_Solid::SetQuantity(INOUT Environment *ev,
IN short howmany) { fQuantity = howmany;}
void MSolid_som_Solid::SetUnitPrice(INOUT Environment *ev,
IN short howmuch) { fUnitPrice = howmuch;}
void MSolid_som_Solid::Initialize(INOUT Environment *ev)
     {
     this->GeneralItem_som_Item::Initialize(ev);
     SetProductOrService(ev, 1);
     SetUnitPrice(ev, 15);
     }
long MSolid_som_Solid::CalcTheATPrice(INOUT Environment *ev)
     {
     SetTheBTPrice(ev, fQuantity * fUnitPrice);
     return(this-
>GeneralItem_som_Item::CalcTheATPrice(ev));
     }


{
MSolid_som_Solid* theSolid = new MSolid_som_Solid;
theSolid->Initialize(ev);
theSolid->SetQuantity(ev, 10);
thePrice = theSolid->CalcTheATPrice(ev);
printf("theSolid Price = %ld\n\n", thePrice);
theSolid->Uninitialize(ev);
delete theSolid;
}
```

As you can see, the Initialize method first calls its parent GeneralItem_som_Item::Initialize, and then calls methods which are defined only in the parent (such as SetProductOrService) or in this class (such as SetUnitPrice) in the same way.

Here's the predictable result:

```
theSolid Price = 162
```

As we see, the calling application can as easily use methods defined by the class (SetQuantity) as methods defined by the parent (CalcTheATPrice) in the same way. So, basically, you design and use inheritance the same way as you would in C++ or Object Pascal or any other OOL. The only big difference is that the methods reside in different shared libraries (SetQuantity is in the SolidSOMLib shared library, whereas CalcTheATPrice is in the ItemSOMLib shared library), although they're being called the same way. This mechanism is transparent both for the developer and for the user. That also means, as we'll see in more detail further below, that the CalcTheATPrice method can be changed in the parent, without SolidSOMLib being any different, the fragile base class problem is solved.

Under its apparent simplicity, both in design and usage, the main strength of SOM resides in this mechanism, that all the methods of an object don't have to be all in the same shared library. Take five to think about it…

```
for(time=TickCount();TickCount()<time+300;);
```

META CLASSES

A good feature of C++ is that we are able to allocate and initialize (with parameters) a class in one statement such as CRect r(0, 0, 50, 100); or CRect *r = new CRect(0, 0, 50, 100);. Although there is an equivalent mechanism in SOM, it's neither so direct nor so easy, but at least it exists.

We saw in the previous examples that our classes inherit from SOMObject. Well, we can also have a class inherit from SOMClass, which we'll use to allocate and initialize (with parameters) our objects. This class is a Meta Class:

And this leads to the predictable result:

```
CarWash with    wax Price = 200
CarWash without wax Price = 50
```

MULTIPLE INHERITANCE

We first create a SOM class (Attributes_som_Attr) the usual way, then we're going to create another SOM class (MTires_som_Tires) which inherits both from Attributes_som_Attr and from MSolid_som_Solid which inherits from GeneralItem_som_Item.

Listing 8

MCarWash_som_CarWash declaration, *implementation and usage*

```
class MCarWash_som_CarWash;
class M_MCarWash_som_CarWash : public virtual SOMClass
{
public:
    virtual MCarWash_som_CarWash*    CarWashCreate(INOUT
      Environment *ev, IN short withWax);
#if __SOM_ENABLED__
    #pragma SOMReleaseOrder (CarWashCreate)
    #pragma SOMClassVersion (M_MCarWash_som_CarWash , 1, 0)
    #pragma SOMCallStyle IDL
#endif
};
class MCarWash_som_CarWash : public virtual
  GeneralItem_som_Item {
public:
    virtual    void    SetWithWax(INOUT Environment *ev,
      IN short yesOrNo);
// overriding
    virtual    void    Initialize(INOUT Environment *ev);
private:
    short    fwithWax;
#if __SOM_ENABLED__
    #pragma SOMReleaseOrder (SetWithWax)
    #pragma SOMClassVersion (MCarWash_som_CarWash, 1, 0)
    #pragma SOMCallStyle IDL
    #pragma SOMMetaClass (MCarWash_som_CarWash,
      M_MCarWash_som_CarWash)
#endif
};

MCarWash_som_CarWash*
  M_MCarWash_som_CarWash::CarWashCreate(INOUT Environment
    *ev, IN short withWax)
    {
    MCarWash_som_CarWash *obj = new MCarWash_som_CarWash;
    if (obj) obj->Initialize(ev);
    if (obj) obj->SetWithWax(ev, withWax);
    return obj;
    }
void MCarWash_som_CarWash::SetWithWax(INOUT Environment
  *ev, IN short yesOrNo)
    {
    fwithWax = yesOrNo;
    if (fwithWax) SetTheBTPrice(ev, 200); else
      SetTheBTPrice(ev, 50);
    }
void MCarWash_som_CarWash::Initialize(INOUT Environment *ev)
    {
    this->GeneralItem_som_Item::Initialize(ev);
    SetProductOrService(ev, 0);
    }
{
MCarWash_som_CarWash* theCarWash;
M_MCarWash_som_CarWash* theCarWashMetaClass;
theCarWashMetaClass = new M_MCarWash_som_CarWash;
theCarWash = theCarWashMetaClass->CarWashCreate(ev, 1);
thePrice = theCarWash->CalcTheATPrice(ev);
printf("CarWash with    wax Price = %ld\n", thePrice);
theCarWash->Uninitialize(ev);
delete theCarWash;
theCarWash = theCarWashMetaClass->CarWashCreate(ev, 0);
thePrice = theCarWash->CalcTheATPrice(ev);
printf("CarWash without wax Price = %ld\n\n", thePrice);
theCarWash->Uninitialize(ev);
delete theCarWash;
delete theCarWashMetaClass;
}
```

The .hh is not very different. After the usual #include directives, we find:

Listing 9

```
MTires_som_Tires declaration and usage
class MTires_som_Tires : public virtual
Attributes_som_Attr, public virtual MSolid_som_Solid {
public:
                    MTires_som_Tires();
    virtual        ~MTires_som_Tires();
    virtual short BrandNew();
#if __SOM_ENABLED__
    #pragma SOMReleaseOrder (BrandNew)
    #pragma SOMClassVersion (MTires_som_Tires, 1, 0)
    #pragma SOMCallStyle IDL
#endif
};

{
MTires_som_Tires* theTires = new MTires_som_Tires;
                                                // from som_Tires
theTires->Initialize(ev);                       // from som_Solid
theTires->SetQuantity(ev, 4);                   // from som_Solid
theTires->SetUnitPrice(ev, 500);                // from som_Solid
theTires->SetTimeBeforeRotation(ev, 3);         // from som_Attr
theTires->SetProvenance(ev, 17);                // from som_Attr
thePrice = theTires->CalcTheATPrice(ev);        // from som_Solid
printf("theTires Price = %ld\n\n", thePrice);
theTires->Uninitialize(ev);                     // from som_Item
delete theTires;                                // from som_Tires
}
```

Leading to the predictable result:

```
theTires Price = 2160
```

The interesting fact to note about the som_Tires object is that although we see it as one object, its code and methods actually reside in 4 different shared libraries: TiresSOMLib, AttrSOMLib, SolidSOMLib and ItemSOMLib, and it's all transparent.

I advise those who are interested in knowing how SOM resolves the classic ambiguities problem when dealing with multiple parents who have different methods with the same name to read chapter 4.2 (Inheritance) of the Users Guide, since I see no point in repeating here the excellent explanations found there. I'll just say that somehow, SOM deals with the ambiguities.

Let's Reap Some Benefits

Now that we have enough classes to play with, let's see how we gain from the use of SOM.

Let's say that suddenly, the state gets greedy (it happens!), and the sales tax on products goes from 8% to 10%.

Well, no big deal. We just reopen the som_Taxes project, change the line ftax = 8; by ftax = 10;, recompile this SOM shared library, and that's it, we're done, we don't have to do any other building of any other library or application using this lib. We just launch our testing application, and the printed results will show that all products (but not the services) are now 2% more expensive:

```
General Item, service, thePrice = 100
General Item, product, thePrice = 110
```

Good, but we could have achieved the same result with a simple dynamic shared library based on CFM, so in that case, although SOM didn't hinder, it didn't do anything that we couldn't have done in another way.

Since, when prices go up, consumers tend to buy less. The providers have to react in some way; most of the time they offer discounts…

So we go back to the som_Item class, and we're going to update it to version 1.1, introducing the discount concept (to prevent confusion, I have a separated ItemDiscount project, with new d2som_ItemDiscount.xxx files, but the built shared library is still ItemSOMLib):

As usual, we start at the .hh, which is now:

Listing 10

d2som_Item.hh v.1.1

```
class GeneralItem_som_Item : public virtual SOMObject {
public:
                    GeneralItem_som_Item();
    virtual        ~GeneralItem_som_Item();
    virtual void Initialize(INOUT Environment *ev);
    virtual void Uninitialize(INOUT Environment *ev);
    virtual void SetTheBTPrice(INOUT Environment *ev, IN
      long thePriceBeforeTax);
    virtual void SetProductOrService(INOUT Environment
      *ev, IN short kind);
    virtual long CalcTheATPrice(INOUT Environment *ev);
    virtual void SetItemDiscount(INOUT Environment *ev, IN
      short howmuch);
private:
    short fProductOrService;
    long  fBeforeTaxValue;
    short fItemDiscount;
    CalcTaxes_som_Taxes* fTheTax;
#if __SOM_ENABLED__
    #pragma SOMReleaseOrder (CalcTheATPrice,
SetTheBTPrice, SetProductOrService,
                                  Initialize, Uninitialize,
                                            SetItemDiscount)
    #pragma SOMClassVersion (GeneralItem_som_Item, 1, 1)
    #pragma SOMCallStyle IDL
#endif
};
```

As you can see, there are only minor modifications. We introduced a new method SetItemDiscount which we also place in the releaseorder directive, and a new field fItemDiscount, and, very important, we changed the minorversion to 1 (it was 0).

SOM Version Numbers

Since I couldn't have explained it better myself, here's the excerpt from the Users Guide about major and minor version numbers:

"These numbers are checked against the version numbers built into the class library to determine if the class is compatible with the client's expectations. The class is compatible if it has the same major version number and the same or a higher minor version number. If the class is not compatible, an error is raised. Major version numbers usually only change when a significant enhancement or incompatible change is made to a class. Minor version numbers change when minor enhancements or fixes are made. Downward compatibility is usually maintained across changes in the minor version number."

We also modify slightly the .cpp:

Listing 11
d2som_Item.cpp v.1.1 (extract)

```
void GeneralItem_som_Item::Initialize(INOUT Environment *ev)
    {
    fTheTax = new CalcTaxes_som_Taxes;
    fItemDiscount = 15;
    }

long GeneralItem_som_Item::CalcTheATPrice(INOUT
  Environment *ev)
    {
    long theValue = fBeforeTaxValue;
    theValue -= (theValue * fItemDiscount) / 100;
    return(fTheTax->CalcTheTax(ev, theValue,
      fProductOrService));
    }

void GeneralItem_som_Item::SetItemDiscount(INOUT
  Environment *ev, IN short howmuch)
    {
    fItemDiscount = howmuch;
    }
```

We build the new library the usual way, and now we reap the SOM benefits: we don't have to rebuild all the libraries containing the classes which inherit from som_Item (ie. som_CarWash, som_Solid, som_Tires, som_Car, som_Tapes), nor do we have to rebuild our testing application. The simple act of rebuilding the new version of som_Item is enough. If we launch the testing application, we will get the following printed result:

```
General Item, service, thePrice = 85
General Item, product, thePrice = 93

CarWash without wax Price = 43
CarWash with    wax Price = 170

theSolid Price = 140
```

That means that if the whole set of these libraries and testing application was a complete solution installed on your customers' Macintosh, to deal with both the new tax and the discount countermeasure, you just modify and rebuild 2 libraries, and you have only those 2 to send to your customers to update them. This is both a gain in time (we only deal with 2 libraries instead of a whole complex global project), and a gain in cost of goods (we only have to send away 2 libraries instead of the whole application). Whatever ways we could choose for the distribution (floppies, Internet, etc.), it will always be cheaper to send a few small items than one big item.

Versioning

Since we now may have 2 different versions of GeneralItem_som_Item floating in our customers' installed base, let's see how we can deal with it and profit from it.

The more the consumer buys, the greater the discount. Let's say that if a consumer buys 1000 tapes, s/he doesn't expects to pay the price of 1 tape times 1000. Now that we have this neat SetItemDiscount method in our GeneralItem_som_Item class, it would be great to be able to call it from one of its descendants. The problem is that this method doesn't exist in the version 1.0 of GeneralItem_som_Item . So let's create a som_Tapes class dealing with discounts and see how it's done. As usual, the .hh first:

Listing 12
MTapes_som_Tapes *declaration and usage*

```
class MTapes_som_Tapes : public virtual MSolid_som_Solid {
public:
                    MTapes_som_Tapes();
    virtual        ~MTapes_som_Tapes();
    virtual void SetDiscount(INOUT Environment *ev, IN
      short howmuch);
#if __SOM_ENABLED__
    #pragma SOMReleaseOrder (SetDiscount)
    #pragma SOMClassVersion (MTapes_som_Tapes, 1, 0)
    #pragma SOMCallStyle IDL
#endif
};

{
MTapes_som_Tapes* theTapes = new MTapes_som_Tapes;
theTapes->Initialize(ev);
theTapes->SetQuantity(ev, 1000);
theTapes->SetUnitPrice(ev, 5);
thePrice = theTapes->CalcTheATPrice(ev);
printf("theTapes Price (no special discount) = %ld\n",
  thePrice);
theTapes->SetDiscount(ev, 25);
thePrice = theTapes->CalcTheATPrice(ev);
printf("theTapes Price (discount = 25\%) = %ld\n",
  thePrice);
theTapes->SetDiscount(ev, 75);
thePrice = theTapes->CalcTheATPrice(ev);
printf("theTapes Price (discount = 75\%) = %ld\n\n",
  thePrice);
theTapes->Uninitialize(ev);
delete theTapes;
}
```

Let's see how the SetDiscount method is dealt with in the .cpp file.

There are many, many ways to verify at runtime if the version 1.1 of som_Item is present, and thus, if we can use its discount feature.

Just like any good Macintosh developer checks with Gestalt to see if such or such feature is available, instead of checking the system version and then making some assumptions about which feature is there or not (these assumptions, often unjustified, have a way to bite the developer back (or rather her/his customers) at a later date), we can check the availability of any method by just using its name.

Two situations can arise: you have the .hh of GeneralItem_som_Item v.1.1, and this should be the usual situation, or for some strange reason (but strange is real life), you don't have the .hh but at least know the prototype of SetItemDiscount (if you don't, then you're stuck).

Since we use that code in a descendant of GeneralItem_som_Item , and any method of GeneralItem_som_Item is in the scope of the methods of that descendant, we simply write in the first case:

```
if (somResolveByName(this, "SetItemDiscount") != 0)
    this->SetItemDiscount(ev, howmuch);
```

Or, in the second case:

```
typedef void (*funcptr)(SOMObject*, Environment*, short);
funcptr callit = (funcptr)somResolveByName(this,
"SetItemDiscount");
if (callit) (*callit)(this, ev, howmuch);
```

And, you're done. Just launching the testing application you'll get, if GeneralItem_som_Item v.1.1 is present:

```
theTapes Price (no special discount) = 4675
theTapes Price (discount = 25 percent) = 4125
theTapes Price (discount = 75 percent) = 1375
And, if GeneralItem_som_Item v.1.1 is not present:
theTapes Price (no special discount) = 5500
theTapes Price (discount = 25 percent) = 5500
theTapes Price (discount = 75 percent) = 5500
```

Now you can understand why the consumer would be pleased with the version 1.1 if s/he qualifies for the 75% discount.

Some other ways to check for the availability of SetItemDiscount:

Using the somRespondsTo method defined in the SOMObject class (which is your ultimate ancestor):

```
char *theMethodName = "SetItemDiscount";
if (this->somRespondsTo(&theMethodName))
     /* SetItemDiscount available */; else /* it's not*/;
```

Using the somSupportsMethod method defined in the SOMClass class (which is an ancestor of your class):

```
char *theMethodName = "SetItemDiscount";
SOMClass *theClass = (SOMClass *)this->somGetClass();
if (theClass->somSupportsMethod(&theMethodName))
     /* SetItemDiscount available */; else /* it's not*/;
theClass->somFree();
```

If you insist on dealing with version numbers rather than features, then you can test if a direct instantiation of som_Item v.1.1 works:

```
SOMObject *theObject;
theObject = somNewVersionedObject(GeneralItem_som_Item, 1, 0);
if (theObject != 0) {
  /* a majorversion of 1 and a minor version >= 0 has been found */
    somReleaseObjectReference(theObject);
} else /* v.1.x unavailable, should never happen... */;
theObject = somNewVersionedObject(GeneralItem_som_Item, 1, 1);
if (theObject != 0) {
  /* a majorversion of 1 and a minor version >= 1 has been found */
    somReleaseObjectReference(theObject);
} else /* most likely only v.1.0 is there... */;
```

As we saw in the "Major and Minor Version Numbers" side bar, if you ask for n.0, it will succeed if n.x is present, whatever the value of x. If you ask for n.m, it will succeed if n.x is present, with x >= m. It will fail in all cases if only p.x is present, with p ≠ n (except if you ask for 0.0, then it will succeed if any p.x is present).

In my example, if both v.1.0 and v.1.1 are present in the same folder as the testing application, v.1.1 will be chosen by default, since SOM, unless told otherwise, will always load the higher possible version.

If you insist on dealing with version numbers rather than features, then you may also check a parent version with the following code (remember that in this case, although GeneralItem_som_Item may be v.1.1 or v.1.0, MTapes_som_Tapes is v.1.0, so it doesn't do us a lot of good testing the version of MTapes_som_Tapes):

Listing 13
GetThisParentVersion *implementation and usage*

```
short GetThisParentVersion(SOMClass *theClass, char
  *className, long *majorVersion, long *minorVersion)
    {
    short i, result = -1;
    SOMClassSequence parents = theClass->somGetParents();
    for (i=0; (i<parents._length) && (result == -1); i++)
        {
        char *parentName = parents._buffer[i]-
          >somGetName();
        if (strcmp(parentName, className) == 0)
            {
            parents._buffer[i]->somGetVersionNumbers
              (majorVersion, minorVersion);
            result = 0;
            }
        else result =
GetThisParentVersion(parents._buffer[i], className,
  majorVersion, minorVersion);
        }
    if (parents._length) SOMFree(parents._buffer);
    return(result);
    }
...
long majorVersion, minorVersion;
SOMClass *theClass = (SOMClass *)this->somGetClass();
short ok = GetThisParentVersion(theClass ,
  "GeneralItem_som_Item", &majorVersion, &minorVersion);
theClass->somFree();
if (ok == 0) /* the parent was found, so we can test the version numbers */;
else /* the parent was not found, something is really weird here... */;
```

Since SOM offers multiple inheritance, the somGetParents method returns an array containing all the direct parents of the class. There is also a somGetParent method which returns only the leftest parent, but I strongly advise you to ignore that one, because it might well get you in trouble in the future (ie. some SOM classes, developed by other developers and which you are using, which were single inheriting from one parent begin to inherit from many parents).

EXCEPTIONS

Since, according to Murphy's law, if anything can go wrong, it will, and according to some other bright person, Murphy was an optimist... It's really a good idea to check our code against reality. The good news is that we can use C++ native exceptions with Direct-To-SOM compilation, the bad news is that we have to do some effort to make it happen.

You can't throw from a SOM method. It's a compiler limitation. Maybe we'll be able to in future releases of C++ compilers, maybe never. What we can do, though, is to ask the compiler, through the SOMCheckEnvironment pragma, to generate calls to __som_check_ev and __som_check_new, after each call to a SOM method or a SOM object allocation. It's up to you to provide an implementation for those calls. You'll find mine below, but you are welcome to make your own. You get your chance to throw in those routines.

Then, each time we're unhappy with something in a SOM method (not enough memory, disk full, and so on), we just fill the

Environment structure with adequate parameters and return. The second parameter (after ev) is the nature of the exception. In your case, it should always be USER_EXCEPTION, then you can pass a string, and a long (in fact a void *). You can use this last parameter to just pass an integer code (that's what I do in this example since I don't have much need for a more complex setup), but you're welcome to allocate whatever structure you want, fill it, and pass its address as this last parameter (you'll have to deallocate it in __som_check_ev later). The exception mechanism you use is your own; you're free to do whatever you want with it.

Then in the calling code, we just try and catch the usual way.

Listing 14
somSetException, __som_check_new, __som_check_ev, *trying, throwing and catching*

```
void MCar_som_Car::SetAudioSystem(INOUT Environment *ev,
  IN short withOrWithout)
    {
    if ((withOrWithout != 0) && (withOrWithout != 1))
        somSetException(ev, USER_EXCEPTION,
                                        "test
somSetException/MCar_som_Car::SetAudioSystem", (void *)1);
    else fAudioSystem = withOrWithout;
    }

#pragma SOMCheckEnvironment on
extern "C" void __som_check_new(SOMObject*);
extern void __som_check_new(SOMObject *SOMObj)
    { if (SOMObj == 0) { throw(1); }}
extern "C" void __som_check_ev(struct Environment*);
extern void __som_check_ev(struct Environment *ev)
    {
    if (ev->_major)
        {
        char *name = ev->exception._exception_name;
        printf("%s\n", name);
        long trowval = (long)ev->exception._params;
        somExceptionFree(ev);
        ev->_major = NO_EXCEPTION;
        throw(trowval);
        }
    }

{
    try {
       printf("calling theCar->SetAudioSystem(ev, 3).\n");
       theCar->SetAudioSystem(ev, 3);
    }
    catch(long x) {
       printf("Caught the throw %ld in catch.\n", x);
       printf("calling theCar->SetAudioSystem(ev, 1).\n");
        theCar->SetAudioSystem(ev, 1);
    }
}
```

Leading to this predictable result:

```
calling theCar->SetAudioSystem(ev, 3).
test somSetException/MCar_som_Car::SetAudioSystem
Caught the throw 1 in catch.
calling theCar->SetAudioSystem(ev, 1).
```

OTHER INTERESTING SOM CALLS

There are many ways in SOM to obtain information on the SOM objects, their names, their parents, their versions, their methods, method names, and method pointers, etc.

Consider the following routines:

Listing 15
PrintParent and PrintInfoAboutSOMObject

```
void PrintParent(SOMClass *theClass, long level)
    {
    SOMClassSequence parents;
    long i;
    parents = theClass->somGetParents();
    for (i=0; i<parents._length; i++)
        {
        char *parentName = parents._buffer[i]-
          >somGetName();
        printf("parent %ld at level %ld: %s\n", i, level,
          parentName);
        PrintParent(parents._buffer[i], level+1);
        }
    if (parents._length) SOMFree(parents._buffer);
    }

void PrintInfoAboutSOMObject(SOMObject *theSOMObject)
    {
    char *theClassName;
    theClassName = theSOMObject->somGetClassName();
    printf("\nclassName: %s\n", theClassName);
    theClassName = theSOMObject->somGetClass()-
      >somGetName();
    printf("className: %s\n", theClassName);
    PrintParent(theSOMObject->somGetClass(), 0);
    long nbOfMethods = theSOMObject->somGetClass()-
      >somGetNumMethods();
    printf("number of methods: %ld\n", nbOfMethods);
    for(long i = 0; i < nbOfMethods; i++)
        {
        somKernelId theKernelID;
        somId theSomID;
        theSOMObject->somGetClass()-
          >somGetNthMethodInfo(i, &theKernelID);
       theSomID = somConvertAndFreeKernelId(theKernelID);
       char *theMethodid = somMakeStringFromId(theSomID);
        printf("method %ld: %s\n", i, theMethodid);
        SOMFree(theSomID);
        }
    }
```

These functions provide 2 different way to get the class name of the passed object and other information. When called this way:

```
theTires = new MTires_som_Tires;
PrintInfoAboutSOMObject(theTires);
delete theTires;
```

It gives this back this result:

```
className: MTires_som_Tires
className: MTires_som_Tires
parent 0 at level 0: Attributes_som_Attr
parent 0 at level 1: SOMObject
parent 1 at level 0: MSolid_som_Solid
parent 0 at level 1: GeneralItem_som_Item
parent 0 at level 2: SOMObject
number of methods: 31
method 0: SOMObject::somInit
method 1: SOMObject::somUninit
method 2: SOMObject::somDuplicateReference
method 3: SOMObject::somCompareReference
method 4: SOMObject::somRelease
method 5: SOMObject::somFree
method 6: SOMObject::somCanDelete
method 7: SOMObject::somGetClass
method 8: SOMObject::somGetClassName
method 9: SOMObject::somGetSize
method 10: SOMObject::somIsA
method 11: SOMObject::somRespondsTo
method 12: SOMObject::somIsInstanceOf
method 13: SOMObject::somDispatch
method 14: SOMObject::somClassDispatch
method 15: SOMObject::somCastObj
method 16: SOMObject::somResetObj
method 17: SOMObject::somDumpSelf
method 18: SOMObject::somPrintSelf
method 19: SOMObject::somDumpSelfInt
method 20: Attributes_som_Attr::SetProvenance
method 21: Attributes_som_Attr::SetTimeBeforePerish
method 22: MTires_som_Tires::BrandNew
method 23: GeneralItem_som_Item::CalcTheATPrice
method 24: GeneralItem_som_Item::SetTheBTPrice
method 25: GeneralItem_som_Item::SetProductOrService
method 26: GeneralItem_som_Item::Initialize
method 27: GeneralItem_som_Item::Uninitialize
method 28: GeneralItem_som_Item::SetItemDiscount
method 29: MSolid_som_Solid::SetQuantity
method 30: MSolid_som_Solid::SetUnitPrice
```

CONCLUSION

So, if SOM is such a great technology, why isn't Apple using it for every little piece of software distributed to either developers or end users?

There are many good reasons. First, object-oriented programing is not necessary everywhere, it is good for an application or a complete solution, but not always very interesting for system software, where encapsulation, polymorphism or inheritance don't always make sense. And if you're not using object-oriented programing, then you're better off using CFM (which Apple is using for every little piece of software) which provides a good dynamic library architecture. But where OOP does make sense, as in Contextual Menus, then Apple uses SOM.

Depending on what you're writing, you will find yourself in one of those cases:

Table 1
Development Strategy

Style	Language(s)	Runtime Architecture
Procedural	C, Pascal, etc.	CFM
Native Objects	C++	SOM
X-Platform or Distributed	Java	Java

Another reason is that, unfortunately, there is a small price to pay to get the benefits of SOM. Although its dispatch code (when you're calling a method which may or may not be overridden) is quite small (6 PowerPC instructions for a non-overridden method, 12 for an overridden method), it may lead to performance issues if you have too fine a granularity (that means that each method does not do much, so the ratio of useful code and dispatch code is not a good one). Typically, if you define classes where methods do jobs as small as those I've shown in the examples of this article, then if you were to overuse these objects (let's say a billion call of SetQuantity and CalcTheATPrice), you might not get the expected performance, even on the fastest Power Macintosh. That being said, let's not forget that if you're using C++ classes, then you also pay a price for method dispatch which is only marginally smaller than SOM's (1 PowerPC instruction for a non-virtual method, 6 for a virtual method). Furthermore, as my experience showed, when defining utility classes, you define most of them as virtual (so that you can override at your leisure later), although you override a small fraction of them in a particular project. In this case, you get 6 dispatch instructions for both C++ and SOM implementation, and your cost is really 6 additional instructions when you actually do an override in SOM. If your methods (which may contain loops, for instance) are thousands instructions long at runtime, then the cost of using SOM instead of C++ is marginal.

Table 2
C++ versus SOM dispatch code

Case	C++	SOM
not overridded	1 instruction	6 instructions
overriddable (virtual)	6 instructions	6 instructions
overridded	6 instructions	12 instructions

So the choice is between non-OOP versus SOM because, if you're going to use OOP, then the advantages of SOM vs. C++ truly more than compensate the very small amount of extra-time spent in the SOM dispatch code vs. the C++ dispatch code.

And, as a reminder, the advantages of SOM are:

- Language and Development Environment independence
- Shared Dynamic Libraries and Object-Oriented Programing
- Keeping the binary compatibility even when:
    - Adding new methods,
    - Changing the size of an object by adding or deleting instance variables,
    - Inserting new parent (base) classes above a class in the inheritance hierarchy,
    - Relocating methods upward in the class hierarchy
- Multi-platform (Mac OS, MS-Windows, OS/2, AIX, and many others...)

Since you, readers, are more likely to develop applications or complete solutions than system software, then you should really think about using SOM to both clean up the past (with encapsulation) and provide for the future.

Thanks to our technical reviewers Deeje Cooley, Pete Gontier, and George Warner.

Thanks to our technical reviewers Brian Arnold, Paul Black, Geoff Clapp, Mike Rossetti, and Jason Yeo. MT

Developer DEPOT™

Developer Depot PO Box 5200 Westlake Village CA 91359-5200
800/MACDEV-1 • Outside the U.S. & Canada: 805/494-9797 • Fax: 805/494-9798
E-mail: orders@devdepot.com • Web Site: http://www.devdepot.com

- ✓ World renowned customer service
- ✓ Order by phone, E-mail, fax or through our continually updated Web site
- ✓ Hundreds of developer products
- ✓ Satisfaction and lowest price guaranteed

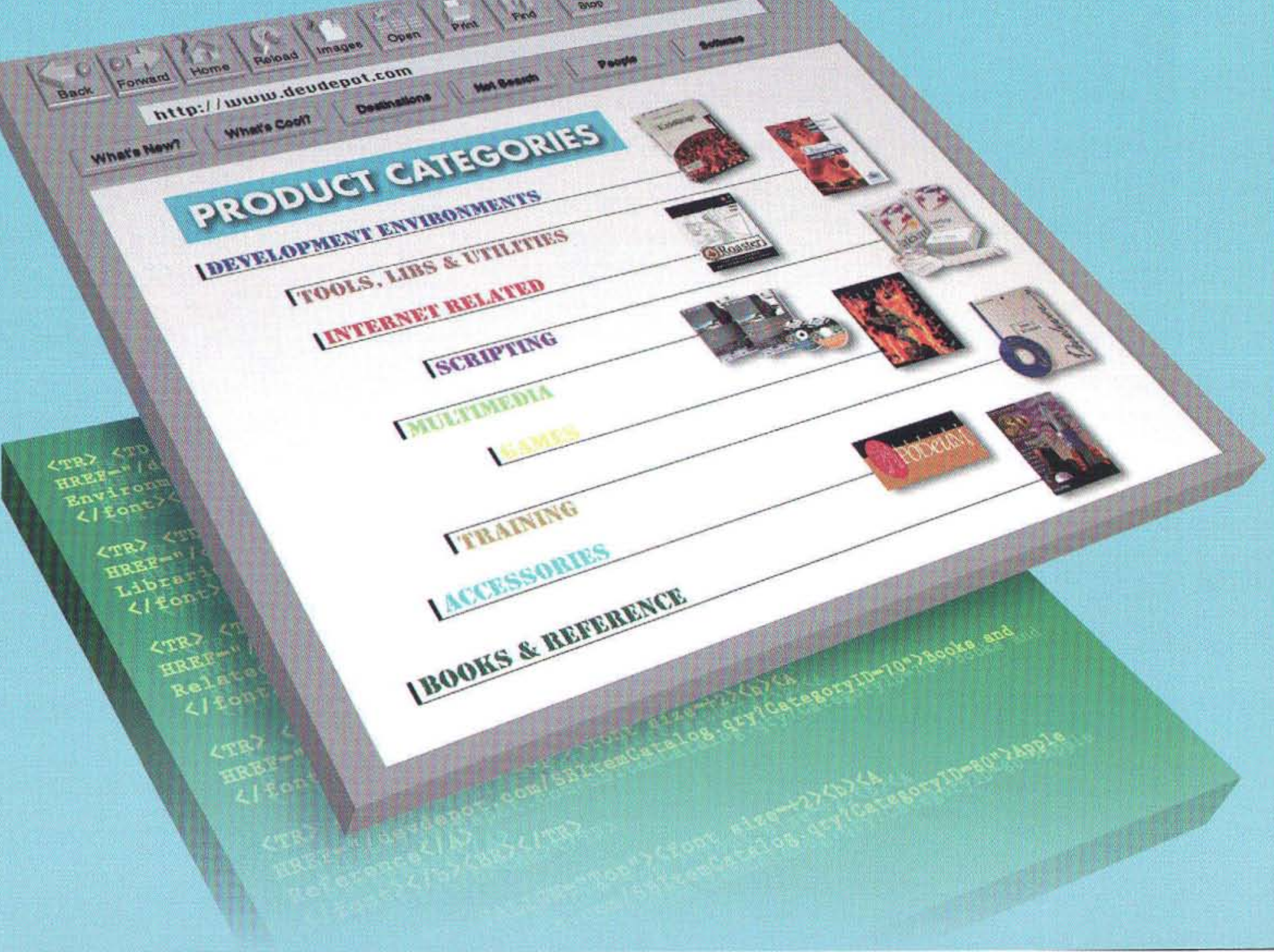


TABLE OF CONTENTS

Lowest price guaranteed. Feel Good about your purchase! 30-day return policy.

One & 2-Day SHIPPING (call for details)

Order Toll-free 800-MACDEV-1 (800-622-3381)

ANNOUNCEMENT

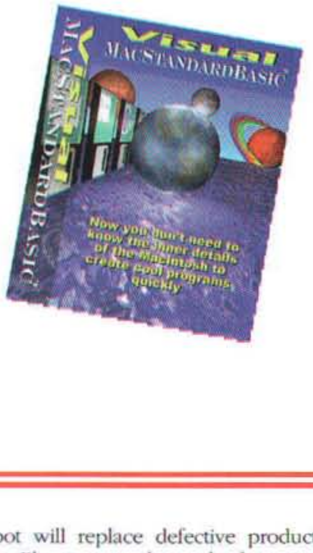
Developer DEPOT™

10% off

Valentine's Day Sale

February 13-February 16

KAIDAN Digital Imaging Technology

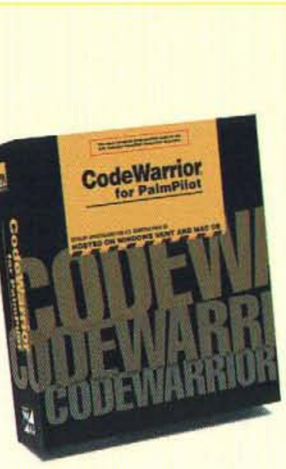

Visual MacStandardBasic
Now you don't need to know the inner details of the Macintosh to create cool programs, quickly

Developer Depot 30 day Money Back, Price and Satisfaction Guarantee

Developer Depot products are sold with a 30 Day money back guarantee on user satisfaction, lower prices and against defects. If, for any reason, you are not satisfied or find the same product at a lower price within 30 days, please call Customer Service at 800-MACDEV-1 and request a Return Merchandise Authorization (RMA) number to get a full refund or the difference in price (where applicable). You must return undamaged product at your expense, including all its original packaging, documentation and the blank warranty card if applicable. Developer Depot will replace defective product upon receipt of the defective merchandise. Please remember to back up your data before installation of any new hardware, software, or peripherals; we cannot be responsible for any lost data. Policies, item availability, and prices are subject to change without notice. The price in effect when we receive your order will be the price that you are charged. We are not responsible for any typographical errors in this or any other catalog, nor for any misstatements from any vendor. Purchase orders are not accepted without prior approval. Call for more information.

Check out our Web site!
• Full product descriptions • Hundreds of more products
http://www.devdepot.com

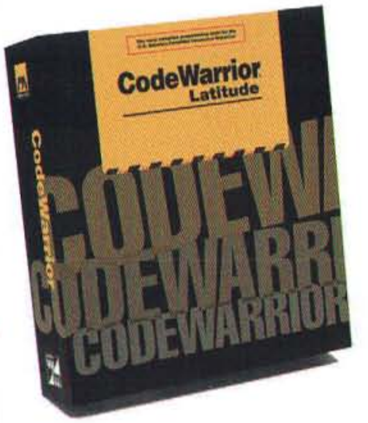
CodeWarrior for PalmPilot
by Metrowerks



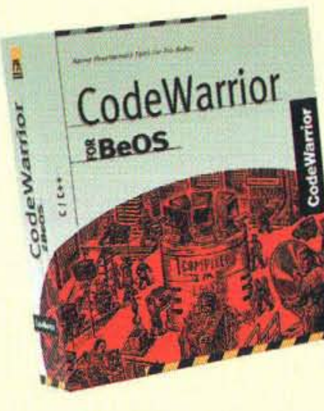
CodeWarrior for PalmPilot is the first complete set of tools which enables you to develop for the U.S. Robotics PalmPilot connected organizer, from the comfort of your PC or Macintosh computer. All the tools you need are included: the award-winning CodeWarrior IDE, compiler, linker, source-level debugger, GUI builder, and other tools, plus online documentation and reference materials. Includes one free product update and free technical support with registration.
(SCWPALM) Our Price **$369**

NEW PRODUCT!

CodeWarrior Latitude
by Metrowerks



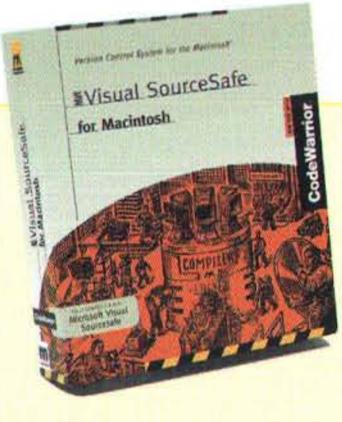
Don't throw away the investment you have made in your Mac OS application! With the new DR2 release of CodeWarrior Latitude, you can now port your Mac OS application to Rhapsody, as well as Silicon Graphics IRIX and Sun Solaris. At the heart of Latitude is a set of shared libraries which performs the functions of the Macintosh API. You recompile your Mac OS source code, linking it with the Latitude libraries to produce a native application. As the Rhapsody API evolves, so will Latitude. Registered users of CodeWarrior Latitude will receive all developer releases, the first full release, plus one additional product update, to ensure that you have access to the most up-to-date Rhapsody porting tools available.
(SCWLAT) Our Price **$399**



CodeWarrior for BeOS 3
by Metrowerks

- Start programming for the new, innovative Be Operating System (BeOS) with complete set of Codewarrior tools
- BeOS-native Integrated Development Environment (IDE) with all the familiar CodeWarrior features at your fingertips
- A BeOS PowerPC compiler and linker, an editor w/syntax color and styling, and a source-level debugger
- BeOS header and libraries, complete documentation, useful C++ classes, and sample code
- Now includes Java support
- The BeOS Preview Release allows you to run and program for the BeOS on a 603 or 604 PCI based PowerMac

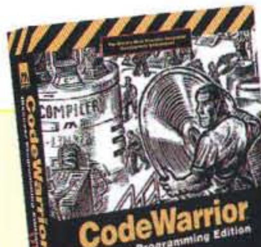
(SCWFB) Our Price **$299**



MW Visual SourceSafe Release 5
by Metrowerks

- Source code control system, plug-in to the CodeWarrior IDE or stand-alone Client application
- Compatible with Microsoft Visual SourceSafe version 5.0
- Cross-platform support (Mac, Windows, UNIX)
- Configuration management in excess of 4 billion files
- Over 8,000 files and sub-projects in a single sub-project
- Registered users receive one year of free technical support and one free update

(SMWVSS) Our Price **$499**



CodeWarrior Discover Programming Edition

by Metrowerks

- Learn to program in C/C++/Java/Pascal
- Full-featured programming product–not a "lite" edition
- Online books and tutorials
- Hosted on either Mac OS or Windows 95/NT

CodeWarrior Discover Programming Edition offers an unbeatable combination of full-featured programming tools, online books and instructional materials, all at a great price. Whether you want to learn the basics of programming or add a new language to your skill set, Discover Programming puts the information and tools you need at your fingertips: the award-winning, easy-to-use CodeWarrior Integrated Development Environment (IDE), four of the most popular programming languages, online books, online tutorials, sample source code and free technical support (with your registration). Discover what you can learn with CodeWarrior, buy your copy today!

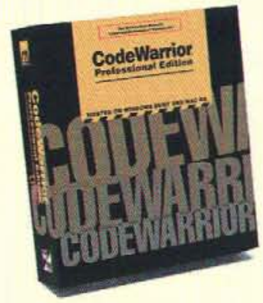
(SDPED) Our Price **$79**

BeOS Preview Release 2

by Be, Inc.

As the Internet spreads and electronic media becomes more prevalent, the high-performance needs of digital content design and the complex, aging architectures of current mainstream operating systems are coming into conflict. The BeOS is the first operating system designed to unlock the door to much more powerful personal computers, and extract more performance from the systems we use today.

(SBEOS) Our Price **$49**



NEW PRODUCT!

CodeWarrior Professional Release 2

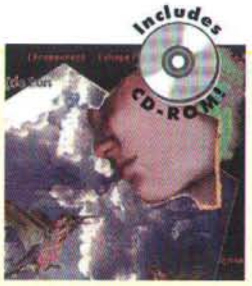
by Metrowerks

- Includes Windows 95/NT and Mac OS versions of the CodeWarrior IDE
- Supports C, C++, Java and Pascal
- Develop for Windows 95/NT on x86 and Mac OS on 68K/PowerPC
- V2 Project Manager supports multiple open projects, subprojects, multiple targets per project, and threaded execution
- Support for JDK 1.1.3 in CodeWarrior Java

More platforms, more languages, more options: CodeWarrior Professional is designed to give you the tools you need for serious, industrial-strength programming. CodeWarrior Professional is the only Integrated Development Environment (IDE) in which you can edit, compile and debug C, C++, Java and Pascal programs for multiple target processors and operating systems. CodeWarrior's compilers produce fast, highly optimized code for Windows 95/NT running on x86, or Mac OS running on 68K or PowerPC processors. CodeWarrior Professional features the CodeWarrior IDE Version 2. The revamped Project Manager supports multiple projects open simultaneously, multiple targets per project, and threaded execution, and it's significantly faster. CodeWarrior Professional also includes online books, documentation, and reference materials, as well as tutorials and sample code. We support your development efforts with one free update and free world-class technical support for a year with registration.

(SCWPRO) Our Price **$449**

Order Toll-free
800-MACDEV-1
(800-622-3381)



Macintosh Common Lisp 4.0

by Digitool, Inc.

Macintosh Common Lisp provides users with a rich set of object-oriented dynamic language features making it especially well-suited for rapid prototyping, custom development for business and education, scientific and engineering applications, and academic research.

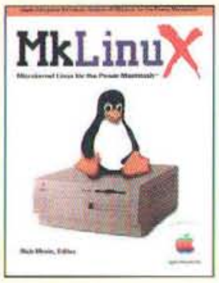
- Power PC native environment & compiler, full Macintosh support
- CLOS, the standard Common Lisp object system
- Interactive dynamic environment, multiple processes
- Automatic memory management and self-typing data
- Ephemeral garbage collector, smaller application footprint
- Compiles with Common Lisp industry standard and smart programmable tools, 110+ mb of user contributed code
- Complete on-line documentation (manual sold separately)
- Software license and registration card

(SMCLISP) Our Price **$675**

Check out our Web site!

• Full product descriptions • Hundreds of more products

http://www.devdepot.com



MkLinux: Microkernel Linux for the Power Macintosh

by Prime Time Freeware

MkLinux is a native port of Mach 3 and the Linux 2.0 kernel, complemented by hundreds of commands from BSD, GNU, and X11. It runs on most (NuBus and PCI bus) Power Macintosh systems; Performa, PowerBook, and multiprocessor ports are currently under development. MkLinux is robust, powerful, freely distributable, and source code compatible with most other Linux systems. It provides a full suite of development tools, support for AppleTalk, HFS, and Objective-C, and access to a vast amount of free software. MkLinux is a great way to "come up to speed" on Mach, UNIX, and Rhapsody.

- MkLinux user community supports FTP and web servers, development and porting efforts, and several mailing lists
- The Apple sponsored reference release contains a wealth of introductory and reference material on Linux, Mach, NeXT, and the Power Macintosh
- Includes free 3.0 upgrade
(BMKLINUX) Our Price **$49**

VIP–BASIC: Visual Interactive Programming in BASIC

by Mainstay

Now you can create full-featured, stand-alone Macintosh and Power Macintosh applications in standard BASIC code! VIP-BASIC 2.0 is the fastest way to program your Macintosh.

- Rapid application development environment with application framework, mix and match: VIP-BASIC high-level subprograms
- Import pre-existing BASIC code: automatically integrate BASIC code, export C Code for compiling: automatically convert your BASIC code to C for compilation with Metrowerks' CodeWarrior
(SVIPBASIC) Our Price **$195**

VIP–C: Visual Interactive Programming in C

by Mainstay

Now you can create full-featured, stand-alone Macintosh and Power Macintosh applications in just minutes. VIP-C 2.0 is the first rapid application development system for creating complete Macintosh programs in standard ANSI C.

- Includes powerful, tightly integrated visual debugger, Import pre-existing C code: automatically integrate C code with a current project
- Includes full-featured mini database: (ltd to 32K) of the powerful VIP-BASIC database manager gives you everything you need to setup royalty-free, multi-user database applications
(SVIPC) Our Price **$295**

Order Toll-free
800-MACDEV-1
(800-622-3381)

Pro Fortran

by Absoft Corporation

Absoft Pro Fortran combines native F90, VAX compatible F77, and C/C++ compilers into a single, easy to use environment. All compilers are link compatible and operate through a common interface.

- Graphical debugger, browsers, array display, performance profiler, linker, MRWE application mainframe
- MIG graphics library, Absoft Create Make, several utilities, the latest version of MPW and illustrated documentation
- Whole array operations, modules, interface blocks, and user-defined types or data structures
- Dynamic memory allocation and new control constructs
- F90 is link compatible with Absoft F77, C++, MrC and CodeWarrior
- It is fully compatible with Toolbox, MPW tools, and most third-party products
(SAPROF) Our Price **$899**

SmalltalkAgents for Macintosh

by Quasar Knowledge Systems, Inc.

- An Integrated Development Environment (IDE) based on QKS Smalltalk.
- "Live" direct manipulation of your objects
- Dynamic, interactive and iterative development process
- Easy and full access to the features of the Mac OS(tm) and Mac Toolbox
- Link your non-Smalltalk code fragments with Code Fragment Manager (CFM) support
- Cross-reference, access, view, and manipulate your code and objects with a sophisticated database for source code management
- Includes an Application Delivery Toolkit(tm) (ADT) that allows you to create royalty-free, standalone, double-clickable applications
(SSTA) Our Price **$395**

Check out our Web site!
• Full product descriptions • Hundreds of more products
http://www.devdepot.com

ObjectMaster Professional Edition

by Altura Software, Inc.

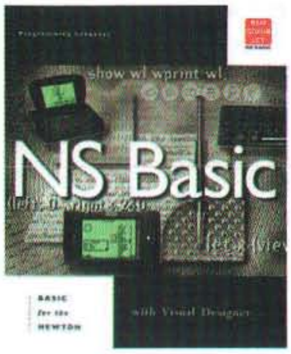
Object Master is an innovative programming environment that provides all the necessary tools to write, organize, and navigate through source code.

- Write code using the most robust source code editor available on the desktop
- Organize source code into projects to quickly access and manipulate all files
- Navigate through source code using intuitive graphical Browser windows

(SOMPE) Our Price **$399**

NS BASIC 3.6 for the Newton with Visual Designer

by NS BASIC Corporation



- A fully interactive implementation of BASIC programming language
- Runs entirely on the Newton – no host is required
- Create files, access the built in soups, and the serial port for input and output
- Work directly on the Newton, or through a connected Mac/PC and keyboard
- Get the **BASIC Internet Tool**, available at no charge to NS BASIC users from www.nsbasic.com
- Release Notes with sample code are available from the same location
- Runs on any Newton MessagePad 130 with NS BASIC and the Newton Internet Enabler. Also runs on MP 1201s with NOS 2.0 that have full memory available
- Write short programs to access News, mail and the web

(SNSBASIC) Our Price **$99**

TENON INTERSYSTEMS

CodeBuilder

by Tenon Intersystems

CodeBuilder is a powerful and unique Macintosh software development tool for porting existing apps or developing new, advanced applications on Power Macs and Power Mac clones.

- A powerful Macintosh software development tool suite of C, C++, Objective-C, Java, Ada, and Fortran development tools
- Complete UNIX & X development environment for developing UNIX or Macintosh apps
- Includes compilers and source-code debugger for Objective C, and C, C++, Ada 95 and Fortran 77
- Web & internet scripting tools: Perl, MacPerl, tcl/tk, bash, sh, and csh
- Supports Rhapsody kernel APIs and Rhapsody TCP sockets

(SMIOCODEB) Our Price **$149**

Order Toll-free
800-MACDEV-1
(800-622-3381)

WAIT... There's More!

Here's a list of all available products. For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us.

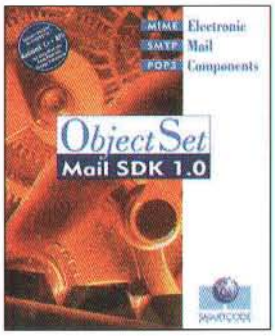
PRODUCT	CODE	OUR PRICE
LPA MacProlog Developers Edition	SLPAD	995.00
LPA MacProlog Programmers Edition	SLPAP	495.00
LS Fortran Pro	SLSFORT	595.00
LS Fortran Plug-In	SLSFPI	199.00
Mac FORTRAN II	SFORT2	595.00
Power MachTen–UNIX	SM10PPC	695.00
Presenting Magic Cap	BPRESMAGIC	15.25
Think Pascal 4.0	SPASCAL	165.00

PowerKey Pro Model 200
by Sophisticated Circuits

PowerKey Pro Model 200 lets you start up and shut down your Mac and up to five peripherals with a single keystroke. Two groups of switched outlets let you control some peripherals separately. PowerKey also features phone ring startup which lets you access your Mac while on the road. Powerful scheduling features let you control your outlets with "hot keys" or perform tasks unattended. Start up your computer at any time of the day or night, open applications and run AppleScripts or QuicKeys. Add the optional Server Restart Option and you can even restart crashed servers automatically!
System Requirements: Mac with ADB port, System 7 or later. Telephone features require analog phone line.
(HPKEY2) Our Price **$99**

PowerKey Pro Model 600
by Sophisticated Circuits

PowerKey Pro Model 600 is "the world's smartest power strip!" Start up and shut down your Mac and peripherals with a single keystroke. Includes six individually-switched outlets, with manual switches and indicator lights. Powerful scheduling features let you control outlets with "hot keys" or perform tasks unattended. Start up your computer at any time of the day or night, open applications and run AppleScripts or QuicKeys. Complete telephone controllability lets you start up the computer, switch outlets or run complex events using custom touch-tone commands. For a limited time, Model 600 includes the Server Restart Option. Restart crashed servers automatically!
System Requirements: Mac with ADB port, System 7 or later. Telephone features require analog phone line.
(HPKEY6) Our Price **$199**



ObjectSet Mail SDK
by Smartcode Software

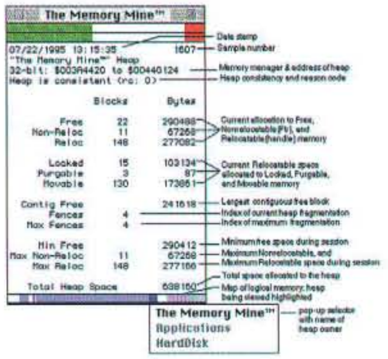
- Powerful C++ classes for integrating Internet e-mail in your applications
- Helps you write software that can share mail with other leading e-mail products
- Royalty-free MIME, SMTP, and POP3 APIs for Macintosh, Windows, and Unix
- Gives you the most robust MIME parser and encoder available
- Ideal for use in Internet and Intranet environments
- Comes complete with samples with documented, reusable source code
- Free standard technical support

(SOSMSDK) Our Price **$495**

NetMinder Ethernet
by Neon Software

NetMinder Ethernet is a software-only protocol analyzer which captures and decodes a full range of Ethernet protocols including IP, AppleTalk, NetWare, NetBIOS and DECnet. Features include:

- Sophisticated long-term monitoring with HTML output
- Intuitive and powerful filtering capabilities
- Automatic mapping of names to Ethernet, AppleTalk, and IP Addresses
- Rules-based engine for detecting unusual network conditions
- Customizable graphs for bandwidth utilization and packet rates
(SNETMD) Our Price **$715**



```
The Memory Mine™
07/22/1995 13:15:35                1607
"The Memory Mine™" Heap
32-bit: $003A4420 to $00440124
Heap is consistent (rc: 0)

                  Blocks      Bytes
         Free        22      290488
    Non-Reloc        11       67268
        Reloc       148      277082

       Locked        15      103134
    Purgable          3          87
      Movable       130      173861

  Contig Free                241618
       Fences         4
   Max Fences         4

     Min Free                290412
Max Non-Reloc        11       67268
    Max Reloc       148      277166

  Total Heap Space           638160

The Memory Mine™
Applications
HardDisk
```

Date stamp
Sample number
Memory manager & address of heap
Heap consistency and reason code
Current allocation to Free, Nonrelocatable (Ptr), and Relocatable (handle) memory
Current Relocatable space allocated to Locked, Purgable, and Movable memory
Largest contiguous free block
Index of current heap fragmentation
Index of maximum fragmentation
Minimum free space during session
Maximum Nonrelocatable, and
Maximum Relocatable space during session
Total space allocated to the heap
Map of logical memory: heap being viewed highlighted
pop-up selector with name of heap owner

Memory Mine
by Adianta, Inc.

- Monitor heaps, identify problems such as memory leaks, and stress test applications
- Active status of memory in a heap is sampled on the fly: allocation in non-relocatable (Ptr), relocatable (Handle) and free space is shown, as are heap corruption, fragmentation, and more
- Allocate, Purge, Compact, and Zap memory lets users stress test all or part of a program

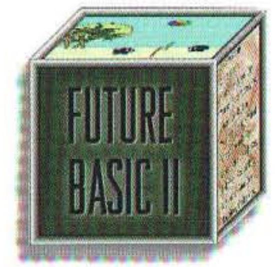
(SMEMMINE) Our Price **$99**

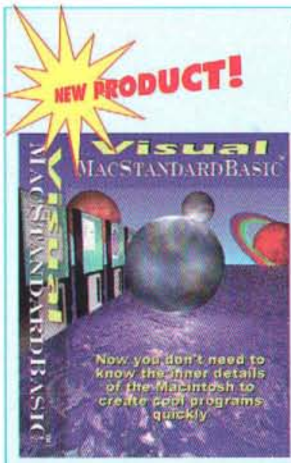
Future BASIC II
by Staz Software

FutureBASIC II is the award winning leader in Macintosh BASIC programming.

- Source level debugger and Interactive compiler/editor
- Multi-file Project manager and Multi-file find and replace
- Super fast compilation, 32 bit clean, and System 7.x savvy
- QuickBASIC converter
- Getting Started manual with over 500 example files
- Full support of standard BASIC

(SFBASIC2) Our Price **$229**




NEW PRODUCT!
Visual
MacStandardBasic
Now you don't need to know the inner details of the Macintosh to create cool programs quickly

Visual MacStandardBasic 3.0
by ZCurve Software

Visual MacStandardBasic is the new standard for creating both 68K and Power Macintosh applications.

- Applications can be visually created in minutes
- Visual controls such as command buttons, text boxes, list boxes, radio buttons, check boxes, scrollbars, icons, pictures and timers can be created and modified instantly
- Use color graphics, animations, movies, sounds and speech in your programs
- Console text window option helps converting older BASIC source code from other platforms
- Online tutorial, manuals, sample projects get you programming quickly

(SVMACSB) Our Price **$29.95**

Spotlight
by Onyx Technology, Inc.

Spotlight is a stand alone debugging aid that performs memory protection (arrays, heap accesses, outside your heap, low mem, etc), discipline checking on toolbox calls, and leaks detection.

- Spotlight is sold on an annual subscription basis
- The subscription service provides all updates
- Includes maintenance releases for one year after the initial purchase or renewal date.

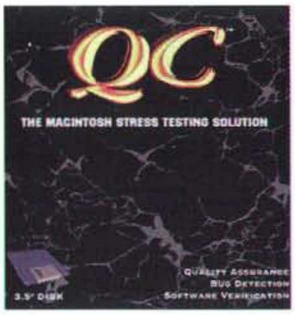
(SSPTLT) Our Price **$199**

QC
by Onyx Technology, Inc.

High performance runtime stress testing for applications.

- Tests include heap checks, purges, scrambles, handle/pointer validation, dispose/release checks, write to zero, de-reference zero as well as other tests like free memory invalidation and block bounds checking
- Extremely user friendly – ideal for non-programmer testers
- Also available in Japanese

(SQC) Our Price **$99**



Celestin

Apprentice 7
by Celestin Company

Apprentice 7 is a high-quality CD-ROM collection of over 600 megabytes of up-to-date source code, utilities, and info for Mac programmers. All of the source code and utilities are completely new or updated for this release.

- Frontier 4.1, the highly-acclaimed scripting environment
- More PowerPlant AND many more PowerPC samples
- Cool new languages and environments added (Clean, Eiffel, F, Tcl-Tk)
- Hot new demos from leading Mac development companies

(SAPPRENT) Our Price **$35**

MacA&D 6.0
by Excel Software

- Structured analysis and design
- Object-oriented analysis and design
- Real-time and multi-task design
- Data and screen modeling
- Integrated code editing and browsing
- Multi-user dictionary and requirements
- Code to design diagrams for C, C++,etc.
- Design diagrams to code for C, C++, etc.
- State modeling diagrams and tables
- Use cases with traceability

(SMACADP) Our Price **$1995**

StoneTable 68K/PPC
by StoneTablet Publishing

StoneTable is a powerful and professional replacement for the List Manager used by developers worldwide. Version 3.0 is a new release with many improvements including better clipboard and drag/drop integration with other applications.

- Available for use with CodeWarrior C & Pascal
- Includes libraries for 68K (A4 & A5) and PowerPC
- An LTable-like class is provided to incorporate StoneTable into the PowerPlant environment

(SSTONEFAT) Our Price **$199**

VOODOO 1.8
by UNI SOFTWARE PLUS

- Stand-alone version control tool for all sorts of projects (software development, documentation, design, CAD, publishing, etc.)
- Smooth integration with Metrowerks CodeWarrior and BBEdit.
- Simple and clear management of variants and revisions of entire projects (not only of single files)
- Easy-to-use graphical project browser gives access to all versions that were ever stored.
- Recording of the complete history (who made which changes when and why)
- View differences between versions (not only for text files!)
- Efficient delta storage of arbitrary files (text as well as non-text files) gains savings of 95 % and more
- Administration of users with hierarchical access rights
- Configurable local file locking (Finder flag or 'ckid' resource)
- Scriptable, essential parts PowerPC native

Single license (SVOODOO1) **$229**
2 pack (SVOODOO2) **$359**
5 pack (SVOODOO5) **$799**
10 pack (SVOODOO10) **$1369**
20 pack (SVOODOO20) **$2399**

Additional pricing available on request.

SEE RELATED CATEGORY: Dev. Environments

Be Basics
by BeatWare

Be Basics is the first productivity suite to combine the essential tools you use everyday with the incredible power of the BeOS. Be Basics offers the BeOS developer the ability to:

- Combine text, tables, graphs and pictures to create compelling documents
- Work with multiple files simultaneously without degrading performance
- View all layout, style and content changes as they happen
- Import Microsoft Excel files
- Plug-in third party graphs and filters

Be Basics requires the BeOS, 16 MB of RAM, 2 MB available hard disk space and a 256-color display adapter.

Price includes all major and minor upgrades through version 2.0 via electronic distribution.

(SBEBASIC) Our Price **$69**

CodeBuilder
by Tenon Intersystems

CodeBuilder is a powerful and unique Macintosh software development tool for porting existing apps or developing new, advanced applications on Power Macs and Power Mac clones.

- A powerful Macintosh software development tool suite of C, C++, Objective-C, Java, Ada, and Fortran development tools.
- Complete UNIX & X development environment for developing UNIX or Macintosh apps
- Includes compilers and source-code debugger for Objective C, and C, C++, Ada 95 and Fortran 77
- Web & internet scripting tools: Perl, MacPerl, tcl/tk, bash, sh, and csh
- Supports Rhapsody kernel APIs and Rhapsody TCP sockets

(SMIOCODEB) Our Price **$149**

Guide Composer™ 1.2
by StepUp Software

- Create powerful Apple Guide help systems for any new or existing Macintosh application
- Provides a WYSIWYG development environment: Guide content is developed in Guide windows
- Design topics, phrases, and panels in the same format as the user will use them
- Features are WYSIWYG interface, Topics, phrases, and hierarchical phrases, Coach marks, Fully-Integrated with Apple's Guide Maker (distributed with Guide Composer), compiles scripts automatically, PICTs in Panels, Generated Guide scripts are modifiable
- FREE Update to all registered Guide Composer users. Demo is available at http://www.guideworks.com/

(SGCOMP) Our Price **$99**

SEE RELATED PRODUCTS: AppleGuide Complete, Danny Goodman's AppleGuide Starter Kit, Real World AppleGuide

B-Tree HELPER 2.2
by Magreeable Software

- Inexpensive database engine for Macintosh programmers in C source code
- Uses contiguous fixed length blocks
- Expands the file as necessary and contracts files when possible
- Inserts and deletes keys in one or more B-Trees
- Finds keys equal to, less than, or greater than a given value in a few hundredths of a second
- Finds lists of records whose keys are equal to, less than, or greater than a given value or are in a range of values

(SBTREE) Our Price **$149**

BBEdit 4.5
by Bare Bones Software

BBEdit 4.5 is a powerful, easy-to-learn text and HTML editor that offers developers and HTML authors the ability to build on its core functionality to suite their specific needs through its plug-in architecture and scripting capabilites. This new version includes: a visual table tool that speeds page and site development, contextual menu support for Mac OS 8, improved storage for 'grep' patterns, scriptable HTML authoring preferences and more. It still provides: unparalleled searching muscle with support for both 'grep' style and advanced literal searches, the ability to quickly compare differences between files or entire folders, integrated support for Symantec's IDE, Metrowerks CodeWarrior, THINK Reference 2.x, MPW Toolserver and most other environments and a heck of a lot more.

(SBBEDIT) Our Price **$119**

Also see Internet Related. page 12

Step-Up Installer Pack
by StepUp Software

- Package of several Installer "atoms" that let developers incorporate graphics, sounds, file compression and custom folder icons into installation scripts
- Compression formats supported are Compact Pro & Diamond
- Each atom also available separately
- Compression requires additional licensing

(SINSTALL) Our Price **$219**

ScriptGen Pro
by StepUp Software

- Installer script generator which requires no programming or knowledge of Rez
- Supports StepUp's InstallerPack, StuffIt decompression, Compact Pro decompression, custom packages, splash screens, network installs, and resource installation

(SSCRPTGEN) Our Price **$169**

BeSpecific 3

(Third in a series)
by Adamation

NEW PRODUCT!

The best-selling BeSpecific CDROM series provides the best in BeOS shareware across a wide range of application areas, including:

- Productivity programs
- Programming tools
- Commercial demos
- Latest source code
- Graphics Games

Newsgroup archives (comp.sys.be) Developer mailing list (Be DevTalk) BeSpecific 3 is brimming with useful BeOS Preview Release-compatible software. A "must have" companion for all users of the BeOS.

(SBESPEC) Our Price **$39**

Pilot Attaché Disk 1

(First in a regular series)
by Adamation

NEW PRODUCT!

The Pilot Attaché CDROM, designed for the popular US Robotics Palm Pilot organizer provides you the best in Pilot shareware. Extend your Pilot's capabilities across a wide range of application areas, including:

- Personal productivity tools
- Programming tools
- Utilities
- Newsgroup information
- Games

Fully tested, Pilot Attaché's shareware treasure throve will help you get the most out of your Palm Pilot. When you travel with your Pilot, don't forget your Attaché. Pilot Attaché...your passport to success.

(SPATCHE) Our Price **$29**

Water's Edge Software

Tools Plus libraries + framework

by Water's Edge Software

Easily create compact, fast running, professional looking applications and plug-ins*. Tools Plus lets you create virtually any user interface element with a single routine, and it transparently provides a robust infrastructure to make all your pieces work together as an application.

- Simplifies programming and thins source code
- Automates all standard GUI elements
- Thousands of extras, from floating palettes and tool bars to powerful picture buttons
- Includes numerous 3D grayscale options
- Over 1/2 MB of custom fonts, icons, cursors, and other resources
- Includes SuperCDEFs world-class controls (an $89 value) free

(STOOLCW) Our Price **$249**
CodeWarrior Gold
(C/C++ & Pascal, 68K & PPC)

(STOOLCWB) Our Price **$199**
CodeWarrior Bronze (C/C++ & Pascal, 68K)

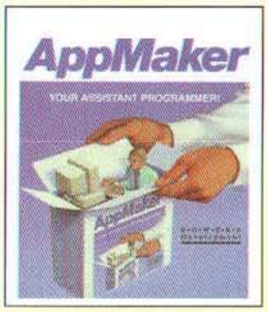
(STOOLSYMT) Our Price **$199**
Symantec (THINK) C/C++ and THINK Pascal (68K)

(STOOLSYM) Our Price **$149**
Symantec (THINK) C/C++ (68K)

(STOOLPAS) Our Price **$149**
THINK Pascal (68K)

*CodeWarrior required to write plug-ins

Lowest price guaranteed. Feel Good about your purchase! 30-day return policy.

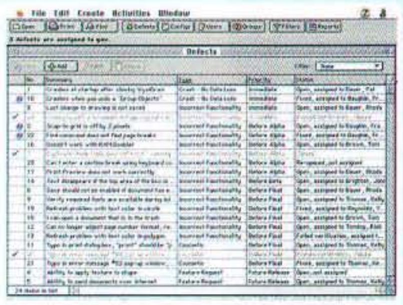


AppMaker

by Bowers Development

- Develop the user interface for a Macintosh application using the original interface builder
- Just point and click to design your application
- Creates resources and generates excellent source code
- Supports most development environments including Metrowerks, Symantec, or MPW; C, C++, or Pascal; procedural or object-oriented, using PowerPlant, TCL, or MacApp
- The generated code uses the Universal Headers to provide PowerMac compatibility
- Great tool for beginners to learn object-oriented and Macintosh Toolbox programming techniques
- Includes one-year subscription on CD and hardcopy documentation

(SAPPMAKE) Our Price **$199**



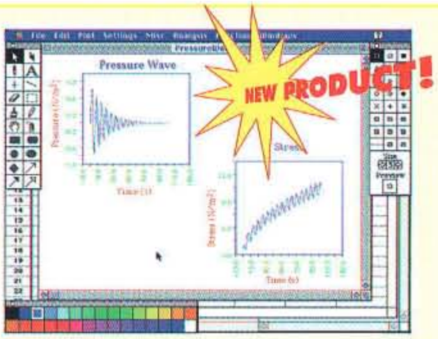
TestTrack-Bug Tracking the Macintosh Way

by Seapine Software, Inc.

- Tracks bugs, feature requests, test configurations, users, and more
- Includes notifications, security, a powerful filter mechanism, and multiple reports
- Links your testers, engineers, documentations staff, and project managers together to ensure all bugs are identified, fixed, and documented
- Eliminates the need to build custom bug tracking solutions using general purpose database tools
- Supports single- and multi-user bug databases (additional licenses required to use multi-user features)

(STETR) Our Price **$129**

SuperAnalyst
by SuperSoft

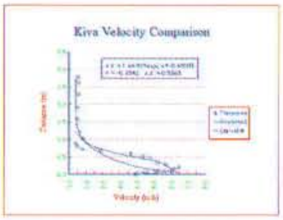


SuperAnalyst is an easy-to-use data analysis and plotting application written specifically for the Apple Macintosh computer and PowerPC. It provides a wide range of X-Y plotting and analysis capabilities at a click of the mouse. It is easy to use and provides interactive control of the appearance of almost every characteristic of your plot. You can overlay multiple plots on the same graph, and the number of points is limited only by the memory of you computer.

- 16 Plot Types: Scatter (Box and Cross), Line (with and without symbols), Function, Log-Log, Semi-Log, Double Y, Bars, Columns, Stacked Bars, Stacked Columns, Area, Pie, Polar, and Histogram.
- Function Plots: Plot mathematical equations you can input without constant reference to the manual. 24 intrinsic functions (cos, sinh, exp, ln, sqrt, erf, bes, gam, etc.).
- PLUS, set templates, reads many file formats, smooth, filter, and sort data, modify data, error bars, function and data integration, FFT tranforms, curve fits, etc.

(SSANAL) Our Price **$99**

SuperPlot
by SuperSoft



SuperPlot is a plotting library which can be linked and called directly from programs written in either Fortran, Pascal, or C. SuperPlot provides a simple way to plot data generated by your program, edit the plot, and then print, export, and/or save the plot for future reference. A simple subroutine or procedure call hides your applications menu bar, puts up SuperPlot's menu bar, creates a new window, and plots your data. It provides you with complete control of the appearance of your plot using the mouse, menus, floating palettes and dialogs. Preparing presentation quality graphs of your data was never easier. SuperPlot runs on either the PowerPC or 68 K computers.

(SSPLOT) Our Price **$195**

NEW PRODUCT!

SuperPlotPRO
by SuperSoft

- Plotting data from your program was never easier
- A Plotting and Chart Library callable from Fortran, Pascal, and C
- Plot Types: Scatter, log-log, x semi-log, y semi-log, Ddouble Y, cross, line, line w/ symbols, bar, stacked bar, column, stacked column, area, pie, polar

(SSPLOTPRO) Our Price **$295**

QUED/M 3.0
by Nisus Software

- The programmer's text editor that defined the industry standard for speed and efficiency
- PowerPC native
- Features integrated support for Symantec C/C++, Metrowerks CodeWarrior 6, and MPW
- Supports all the major development environments on the Macintosh.
- Powerful editing features, including unlimited undo and redo, macro language, scripting, text folding, ten editable/appendable clipboards, markers, displaying text as ASCII codes, dynamic coloring of C/C++ keywords/comments, rectangular and non-contiguous selection
- Includes Celestin Company's APPRENTICE 4

(SQUEDM) Our Price **$89**

AG Author
by Lakewood Software

AG Author 1.0 is a full-featured Apple Guide authoring tool with fully customizable project template. The following features are unique to AG Author:

- Support for styled, colored, & hot text
- Fully customizable project template
- Flexible compile options
- Find & replace tool for scripts
- Multiple open projects
- Rapid deployment of project globals

(SAGA) Our Price **$99**

SEE RELATED PRODUCTS: AppleGuide Complete, Danny Goodman's AppleGuide Starter Kit, Real World AppleGuide

Web Ware
by BeachWare, Inc.

The ultimate collection of clip media and templates for building your own Web Page. An incredible selection of Shockwave movies, animated GIFs, buttons, bullets, dividers, and sample HTML pages. There are literally thousands of graphical elements on this disc, all there to spice up your web page. In all, it's about 300 megabytes of creativity only a mouse-click away! System Requirements: PC - 486 or better with 8 MB RAM, Sound card, SuperVGA, CD-ROM drive. Macintosh - Color Mac with 8 MB RAM, CD-ROM drive.

(SWEBW) Our Price **$24**

SoftPolish CD-ROM
by Bare Bones Software

- The essential tool for software quality assurance on the Macintosh
- Helps you identify inconsistencies with Apple's user interface guidelines, misspelled words, missing resources, and other mistakes
- Provides tools to put the finishing touches on software distribution packages prior to release
- Works independently of any programming language or environment
- Ideal for sanity checking software throughout the development process

(SSOFTPOL) Our price **$99**

VText
by Vivistar

VText is a C++ add-on library for Metrowerks' PowerPlant application framework. VText provides complete Macintosh text support including: greater than 32kb text, undo, drag and drop editing, AppleEvent scripting and recordability, full support for multibyte characters and inline input methods including Japanese and Chinese text, and full support for bi-directional script systems including Arabic and Hebrew.

- Full featured text engine for Metrowerks' PowerPlant
- Stylesets and rulersets with tabs
- Flexible object oriented C++ API
- Full undo and drag and drop editing
- WorldScript savvy including bidirectional and multibyte scripts with inline editing
- AppleEvent factored for scriptability and recordability

(SVTEXT) Our Price **$349**

OpenGL for the Macintosh
by Conix Graphics

OpenGL is the premier 3D graphics library that allows software developers the ability to develop high-quality, interactive 2D and 3D graphics applications. OpenGL can perform the following wide range of functions which will enhance the development of all graphics software:

- Geometric primitives (points, lines, and polygons)
- RGBA or color index mode
- Viewing and modeling transformations
- Texture Mapping, Lighting, Shading and Z Buffering
- Atmospheric Effects (fog, smoke, and haze)
- Alpha Blending (transparency)
- Antialiasing, Accumulation Buffer, Stencil Planes
- Display list or immediate mode
- Polynomial Evaluators (to support Non-uniform rational B-splines)
- Feedback, Selection, and Picking Raster primitives (bitmaps and pixel rectangles)
- Pixel Operations (storing, transforming, mapping, zooming)

(SOPENGL) Our Price **$389**

WAIT... There's More!

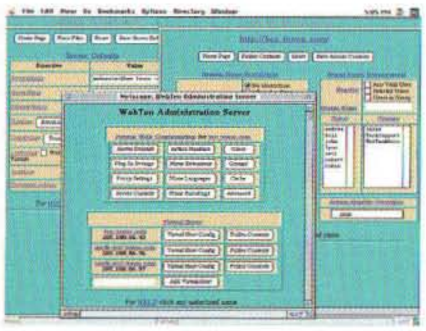
Here are more products. For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us.

PRODUCT	CODE	OUR PRICE
Bee-one	SBEEONE	$139.00
C-tree Plus® Database Handler	SCTPDH	$895.00
CompileIt!	SCOMPIT	$149.00
CPU Doubler	SCPU2X	$79.00
DesignWorks 4.0	SDWORKS	$995.00
dtF	SDTF	$695.00
EtherPeek	SEPEEK	$745.00
Fortran 77 SDK	SF77	$699.00
ICONIX PowerTools-6 Pack	SICPP6	$5,945.00
ICONIX PowerTools-8 Pack	SICPP8	$6,945.00
ICONIX PowerTools-10 Pack	SICPP10	$7,845.00
ICONIX PowerTools-AdaFlow	SICADA	$1,395.00
ICONIX PowerTools-ASCII Bridge	SICASCII	$1,395.00
ICONIX PowerTools-CoCoPro	SICCOCO	$1,395.00
ICONIX PowerTools-DataModeler	SICDATAMOD	$1,395.00
ICONIX PowerTools-FastTask	SICFASTTASK	$1,395.00
ICONIX PowerTools-FreeFlow	SICFREEFL	$1,395.00
ICONIX PowerTools-Object Modeler	SICOBJMOD	$1,395.00
ICONIX PowerTools-PowerPDL	SICPOWER	$1,395.00
ICONIX PowerTools-QuickChart	SICQUICKCH	$1,395.00
ICONIX PowerTools-SmartChart	SICSMART	$1,395.00
ICONIX Training & Consulting	TICONIX	$2,945.00
IMSL Math and Stat Library	SIMSLSTAT	$495.00
Info-Mac X	SINFOMAC10	$39.00
Ionizer Real-Time Spectral Reshaping Tool	SIONIZER	$800.00
LiveAccess™ 1 User Edition	SLAUE	$69.00
LiveAccess™ 1 Developer Edition	SLADE	$99.00
LiveCard	SLCARD	$149.00
LJ Profiler	SLJPROF	$295.00
MacFlow™: Flowchart Design and Development	SMACFLO	$179.00
Mac Source II	SMACSOURCE	$29.95
Nisus Writer 5.0	SNISUSW	$220.00
Plan & Track™: Project Planning and Management	SPLNTRK	$179.00
Phyla™: Object-Oriented Database	SPHYLA	$179.00
r-tree Report Generator	SRTRG	$445.00
Spellswell Plus 2.1	SSPELL	$49.00
Spyer	SSPY	$39.00
Visual Café	SVCAFEMAC	$199.00

Check out our Web site!
• Full product descriptions • Hundreds of more products
http://www.devdepot.com

PageCharmer: Sizzling Effects...

NEW PRODUCT!



WebTen
by Tenon Intersystems

WebTen is an industrial-strength, high-performance Apache Web server for Power Macs. WebTen's Web-based browser interface enables local or remote administration via your favorite browser. Since Apple's NeXT acquistion, Tenon has extended their unique "UNIX virtual machine" technology to produce a set of "Rhapsody-Ready" internet applications. WebTen is the first offering in this series.

- WebTen is the fastest Web server on Power Macintosh
- Sustains up to 10,000 connections a minute, or over 10 million connections a day
- Apache runs in Tenon's multi-threaded, pre-emptive multitasking environment
- Tenon's unique technology supports the widely acclaimed Apache Web server as a double-clickable Macintosh application

(SWEBTEN) Our Price **$495**

PageCharmer 1.0
by Mainstay

PageCharmer is a set of customizable interactive applets that enhance web pages without writing a single line of HTML code. Whether the web site is already up and running or designing one from scratch, PageCharmer gives you the power to make it stand out from the crowd with sophisticated applets that can be personalized to fit most any need.

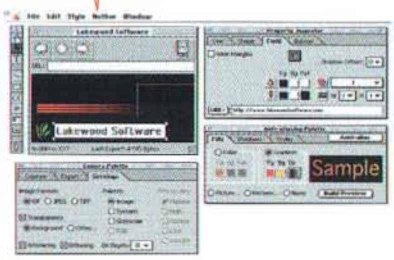
FEATURES:
LiveG-Map, LiveT-Map, LiveG-Button, LiveT-Button, LiveGT-Button, LiveG-Ticker, LiveT-Ticker, LiveG-Marquee, and LiveT-Marquee.
(SPGCHRM) Our Price **$99**

BBEdit 4.5
by Bare Bones Software

(SBBEDIT) Our Price **$119**

Also see Tools, Libraries and Utilities, page 8

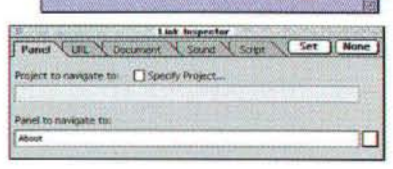
NEW PRODUCT!



webAlias 1.0
by Lakewood Software

webAlias 1.0 is an integrated image map editor and anti-aliasing text tool for web and graphic designers. Use webAlias to create complete web sites, single web pages, and graphic content for multimedia and web design projects. webAlias integrates support for line, shape, free form, field and button objects. webAlias' anti-aliasing features include support for embedded pictures and gradients in text, as well as multiple shadow and highlight effects.
(SWEBALS) Our Price **$129**

NEW PRODUCT!



Founded in 1995, Lakewood Software develops and publishes multimedia and development tools for Macintosh professionals.

Our products include HyperGuide, a hybrid multimedia authoring tool and on-line documentation system, webAlias, an image map editor and anti-aliasing text tool, and AG Author, a full-featured Apple Guide authoring tool.

HyperGuide 1.0
by Lakewood Software

HyperGuide 1.0 is a hybrid multimedia authoring tool and on-line documentation system for the Macintosh and World Wide Web. HyperGuide provides integrated searching, indexing and bookmarking features. Supported media elements include: rectangle and scrolling fields, lines and shape fills, most QuickTime-supported image formats, anti-aliased text and QuickTime VR movies. HyperGuide also includes an integrated screen capture utility and user-configurable slide show mode.
(SHYPGUD) Our Price **$149**

ObjectSet Mail SDK

by Smartcode Software

- Powerful C++ classes for integrating Internet e-mail in your applications
- Helps you write software that can share mail with other leading e-mail products
- Royalty-free MIME, SMTP, and POP3 APIs for Macintosh, Windows, and Unix
- Gives you the most robust MIME parser and encoder available
- Ideal for use in Internet and Intranet environments
- Comes complete with samples with documented, reusable source code
- Free standard technical support

(SOSMSDK) Our Price **$495**

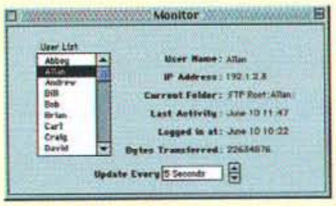
Power MachTen 4.0.3

by Tenon Intersystems

MachTen is the only Macintosh product that can turn your Macintosh into a complete Unix workstation. Based on BSD4.4 and the Mach kernel, MachTen brings the power of Unix to your desktop at an extremely attractive price point. MachTen enables you to:

- Run a high speed internet server, complete with WWW, FTP, NFS, DNS and print service
- Build a Mutihomed Web Server
- Develop applications in a Unix development environment, replete with the acclaimed GNU development toolset
- Program in Ada, C, C++, Pascal, Fortran, and more
- Run Xwindows applications, from remote workstations or on your Macintosh
- Run hundreds of Unix applications, already ported for MachTen and available on our Ported Applications CD-ROM
- Run Software.com Inc's acclaimed Post.Office mail transport service

(SM10PPC) Our Price **$695**



User Name: Allan
IP Address: 192.1.2.8
Current Folder: :FTP Root:Allan:
Last Activity: June 10 11:47
Logged in at: June 10 10:22
Bytes Transferred: 22634876
Update Every 5 Seconds

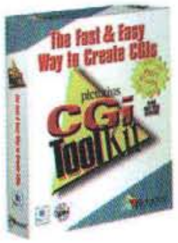
Rumpus

by Maxum Development

Maxum's new, high-performance FTP server for the MacOS. Based on Maxum's RushHour TCP/IP implementation, Rumpus 1.0.1 offers the performance and reliability of high-end workstations with the ease of use, security, and flexibility of the Macintosh.

- Simplified setup, with no need to configure AppleShare, File Sharing, or Users & Groups for simple anonymous FTP
- Anonymous and/or secure server access, with separate security settings for anonymous vs. secure users
- Automatic MacBinary and Binhex encoding
- Complete logging, with separate anonymous and secure access logs, including anonymous user passwords
- Up to 32 simultaneous connections

(SRUMP) Our Price **$195**



CGi Toolkit

by Pictorius, Inc.

The Pictorius CGi Toolkit is the fast and easy route to high performance CGIs and ACGIs for your Mac Web site.

- Interactively develop CGIs while the web server, the CGI Toolkit and the browser are running on the same machine
- Interactively develop, test and debug CGIs before compiling
- Powerful debugger allows you to edit code, roll back, code and change input values while your application is running
- Fully object oriented so you can re-use your code
- Automatic handling of Apple Events so you can concentrate on building functionality
- Easy creation of multi-function CGIs which reduces application footprint and RAM usage

(SCGITLKT) Our Price **$149**

Here are more products. For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us.

PRODUCT	CODE	OUR PRICE
OOFILE Reporter Writer	SOORW	$499.00
ScriptDemon	SSDEMON	$949.00
WebSiphon	SWSIPHON	$495.00

WindowScript
by Royal Software, Inc.

WindowScript is the ultimate tool for designing Macintosh user interfaces using HyperCard. Design Real "Macintosh" user-interfaces right inside HyperCard. Until now you either created HyperCard stacks or Macintosh applications. With WindowScript you can literally bring the look and feel of a real Macintosh user-interface to HyperCard. If you're a HyperCard developer, interface designer, application developer, program manager or tester searching for a prototyping tool, WindowScript is perfect for the job.
(SWSCRIPT) Our Price **$149**

Script Debugger
by Late Night Software Ltd.

- A powerful and flexible AppleScript authoring tool – get the most from AppleScript!
- Advanced debugging environment offers single-step script execution with breakpoints
- Script Debugger dictionary browser features a graphical view of objects provided by scriptable applications
- Includes Late Night Software Scripting Additions – a collection of more than 70 new AppleScript commands, and Scheduler, a utility that allows you to launch scripts at pre-determined times

(SDEBUG) Our Price **$129**

Scripter 2.0
by Main Event Software

For professionals, for novices, for webmasters, for solutions providers, there's only one serious choice. Scripter!

- Scripter and FaceSpan work together: one click opens your FaceSpan script in Scripter, another sends it back
- Debug handlers without modifying your scripts using the Call Box
- Applet simulation, live editing, Object map, associated terminology
- Search backwards, block generators, more navigation shortcuts, more drad-and-drop, and an even more enhanced trace log
- Now Includes ScriptBase; stores your data and media elements and share them between scripts all with a special new browser
- Easily write and compile scripts that have handler declarations and other vocabulary specific to a particular scriptable application
- Scripter is the natural companion to AppleScript for users at all levels of proficiency. Don't write scripts without it!

(SSCRIPTER) Our Price **$199**

NEW PRODUCT!

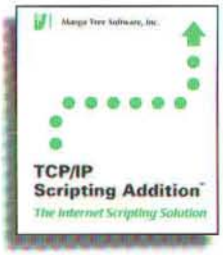
FaceSpan v3.0
by Digital Technology International

FaceSpan is a cutting edge interface design and rapid application development (RAD) tool which gives you the power to build and customize Macintosh applications quickly and easily.

- Acts as your front end for AppleScript or any other OSA (Open Scripting Architecture) language.
- Allows you to automate often-repeated tasks, customize and integrate existing applications, build new applications and personalize your computing environment.
- NEW! Allows you to create interfaces and applications that conform to the Mac OS 8 look and feel.
- NEW! Supported display objects now include tab panels, disclosure triangles, bevel buttons and more.
- NEW! FaceSpan run-time now launches up to 5X faster.
- Includes an unlimited, royalty-free distribution license for the interfaces and applications you create.

(SFACESPAN) Our Price **$149**

TCP/IP Scripting Addition
by Mango Tree Software



- Award-winning AppleScript scripting addition
- Allows you to write scripts using MacTCP™ commands in AppleScript™
- Send e-mail or files through a script, check if users are logged on (via Finger), automate FTP, Gopher, NetNews, Telnet, and LPR, verify links in HTML documents, and quickly write many other TCP/IP client-server programs
- Works with AppleScript, MacTCP 2.0.4 and Open Transport

(STCP) Our Price **$49**

DynaMorph 1.5
by Morph

DynaMorph

DynaMorph is the only cross-platform, server-side scripting language. Easily build and maintain dynamic websites and web-based applications. Access external databases, separate the format of a website from its content, conduct e-commerce transactions and more. DynaMorph makes sites and applications completely portable.
(SDYNA) Our Price **$399**

WAIT... There's More!

Here are more products. For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us.

PRODUCT	CODE	OUR PRICE
PreFab Player	SPLAYER	$95.00

Clip VR

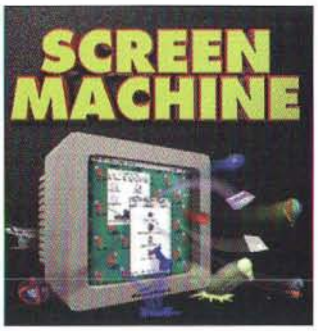
Clip VR™
by eVox Productions

Clip VR™ is a new digital image library offering high quality Photographic Virtual Reality (PVR) images for use with Quicktime® VR and other desktop VR tools.

Clip VR™ Panoramic Image components include alpha channel masks. Combine elements into a composite panorama which is converted to the finished QuickTime VR movie using Make QTVR Panorama Tool. The Components ("Clips") include complete 360 degree scenes, libraries of 360 degree terrains, 360 degree skies, buildings, and objects. Images are provided as .PICT files AND QuickTime VR movie files. Clip VR™ allows you to create high quality VR worlds from pre-photographed component images. Clip VR™ can be used to add excitement to a web site, to increase the interactive value of a CD-ROM, or simply for fun! Requires imaging editing program such as Adobe Photoshop™ to perform .PICT image compositing.

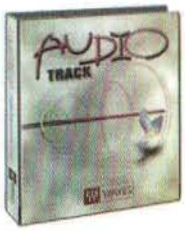
(SCLIPVR) Our Price **$89**

Screen Machine
by BeachWare, Inc.



Personalize your computer screen with this dynamic and useful collection of Screensavers and Wallpapers. Choose from over 100 original Screensavers such as flying airplanes, bouncing coffee cups, falling climbing gear, shifting psychedelic patterns, or floating spaceships. Customize your computer desktop with over 150 exciting new wallpapers such as sand, rocks, cloth, coins, cartoons, or unique patterns. Included is an easy to use browser program that lets you sample all of the Screensavers and Wallpapers before installing them on your computer.

(SSM) Our Price **$24.95**



AudioTrack
by WAVES

AudioTrack is a software plug-in for native processing on digital audio recording and editing systems. Waves AudioTrack combines the most-needed audio processors into a single piece of software, including 4 bands of equalization, compression/expansion, and noise gating. AudioTrack is ideal for preparing audio for InterNet streaming formats, processing individual mono/stereo tracks of audio and audio for video editing systems including digital sequencers.

Feature list
- A single window interface
- 4-band ParaGraphic Equalizer, Compressor/Expander and Gate
- Instantaneous A/B comparisons of on-line settings
- Pretested setup libraries supplied for various processing solutions
- Power Macintosh native processing (requiring no DSP board)
- Volume and gain reduction meters
- Peak hold and clip meters

Requirements:
The AudioTrack is compatible with all machines supported by Deck II, SoundEdit 16, Adobe Premiere 4.0 and Cubase VST 3.1

(SAUDIOTRK) Our Price **$270**

Captivate 4.6: Essential Graphics Utilities
by Mainstay

Captivate™ 4.6 is a powerful collection of graphics utilities for Macintosh, based on Mainstay's acclaimed screen capture utility, graphics and multimedia scrapbook, and graphics viewer. Captivate provides essential graphics utilities to the professional and hobbyist alike.

- Package includes: Captivate Select, Captivate View, and Captivate Store
- Any one of the three can be used alone, and together they make an unbeatable team
- Whether writing a training manual, creating an ad, or just creating a startup screen from your favorite picture, Captivate is everything professionals need at a price anyone can afford.

(SCAPTIV) Our Price **$79**

Lowest price guaranteed. Feel Good about your purchase! 30-day return policy.

Media Cleaner Pro
by Terran Interactive

Use Media Cleaner Pro 2.0 to optimize and compress video for CD-ROM, kiosk, or the Internet. Media Cleaner Pro automates your work flow allowing you to get the highest quality video, faster and easier than any other program on the market.
- Includes Adobe Premiere Export module
- Optimal palette generation, Drag-and-drop batch processing
- RealMedia, VDOLive and improved QuickTime support
- Dynamic Preview Window, the Media Wizard, multiprocessor support and more!

System Requirements:
68040 Mac or better (PowerPC strongly recommended, req'd for RealMedia), QuickTime 2.0 or later (2.5 strongly recommended)
8 Mb application RAM, MacOS 7.0.1 (7.5 or later recommended)
SoundManager 3.2, CD-ROM Drive

(SMCP) Our Price **$359**

Registered owners of Movie Cleaner Pro 1.3 or earlier can upgrade

(SMCPUP) Our Price **$129**

KAIDAN Digital Imaging Technology

Magellan QC
by Kaidan

The Magellan QC is capable of handling objects as large as six inches in diameter and five pounds in weight, the Magellan QC is the perfect choice for those needing to capture small objects at a reasonable price. Real-world objects can be turned into 3-D virtual reality movies using the QuickTime VR Authoring Studio and the Magellan QC.

The Magellan QC leverages the capabilities of the Connectix™ Color QuickCam™ digital camera for QTVR object capture. The Color QuickCam's close focusing capability (one inch to infinity), 640 x 480 resolution, serial interface (no video card required), 24-bit color support, convenient size and low cost make it an ideal camera for many simple QTVR object movies. Using the Magellan QC is easy. Simply locate the object on top of the adjustable pedestal, perhaps with a small piece of double-sticky tape, and then adjust the arms and pedestal so that the center of the object is centered in line with the camera and the rotation axis of the swingarm.
(HMAGQC) Our Price **$299**

Magellan Accessories

Magellan QC Pedestal Set
Two extra pedestal tube assemblies, one 2.5" and another 6" long. These extra pedestal tubes are used to support objects of varying sizes on the Magellan QC.
(HMAGPED) Our Price **$39**

Magellan QC Detent Wheels
A pair of optional detent wheels (Color = Gold) with 8 (45 deg), 12 (30 deg), 14 (25.7 deg), 16 (22.5 deg) and 18 (20 deg) settings. The standard wheels (Color = Aqua) provide 10, 15, 20, 24 and 36 positions.
(HMDWHLS) Our Price **$74**

QuickPan Magnum
by Kaidan

The QuickPan Magnum Series consists of two models, the QPX-1 and QPX-2. Featured on both models is the new QPU-2 camera bracket. Based on the highly successful KiWi, it provides a sturdy, collapsible system for the mounting and adjusting of a wide variety of cameras and camcorders. The new base designs used on the Magnums are a refinement of our earlier bases, with the QPX-1 having a fixed base and the QPX-2 having a new low-profile micro-tilt adjustment stage. The easily adjustable click-stops will let you capture a panorama in a few seconds. The QPU-2 has two accessories, a Landscape Bracket for positioning the camera in the landscape orientation (QPLB-1) and a Counterweighting Kit (QPCW-1) used to balance large cameras or camcorders, such as the Sony VX-1000, that have a center of mass well behind the pivot axis.

QuickPan Magnum-1	(HQPMAG1)	Our Price **$499**
QuickPan Magnum-2	(HQPMAG2)	Our Price **$549**

QuickPan Magnum Accessories

Counterweighting Kit
The Counterweighting Kit includes a weight and adjustable arm that is used to offset the weight of large, heavy cameras and camcorders.
(HWHTKT) Our Price **$129**

Detent Wheel
Detent Wheel (5-inch) (Color = Purple): 10, 14, 18, 24 and 30 Position (QPDD-2)
(HQDWHLS) Our Price **$49**

QuickTilt Leveler
A leveling stage, similar to the one found on our QuickPan Magnum QPX-2, that mounts between your panhead or camera and your tripod. It makes the leveling process quick and easy. Particularly useful when you plan to shoot a number of QTVR/VR nodes in a short period of time.
(HQTLVLR) Our Price **$149**

KiWi
by Kaidan

The KiWi™ is the most affordable VR/QTVR panhead, bringing digital photographic panoramas to an even wider audience. It's the perfect companion to programs such as QuickTime VR Authoring Studio, PhotoVista and Nodester, providing a complete solution for anyone interested in adding VR panos to their websites and multimedia applications. The KiWi™ consists of two intersecting black anodized aluminum struts that adjust and lock to accommodate a wide range of cameras, such as the Apple QuickTake 100/150/200, Kodak DC50/120, APS film cameras and 35mm SLRs equipped with wide-angle lenses. KiWi™ attaches to any standard tripod and camera equipped with a standard 1/4-20 mounting thread.

(HKIWI) Our Price **$99**

One & 2-Day SHIPPING (call for details)

KiWi+
by Kaidan

The KiWi+ adds a compact, yet durable click-stop mechanism and the same twin-axis bubble level found on the top-of-the-line QuickPan Magnum Series heads. The twin-axis bubble level (recommended by Apple and VR professionals) provides a clear indication of level, even when the unit is slightly above eye level. The click-stop mechanism uses easily replaceable detent discs, which are available in a number of positions (8, 12, 16, 18, 20). The KiWi+ ships with one disc of your choice and extra discs are available separately or as a set. The click-stops speed the process of shooting a panorama by eliminating the need for the photographer to look at the unit in order to visually align the index increment.

(HKIWIP) Our Price **$249**

KiWi and KiWi+ Accessories

QuickTilt Leveler
A leveling stage, similar to the one found on our QuickPan Magnum QPX-2, that mounts between your KiWi or KiWi+ and your tripod. It makes the leveling process quick and easy. Particularly useful when you plan to shoot a number of QTVR/VR nodes in a short period of time.

(HQTLVLR) Our Price **$149**

KiWi-to-KiWi+ Upgrade
Includes the necessary parts required to turn your KiWi into a KiWi+ — adding click-stops and the twin-axis bubble level. Comes with a detent disc of your choice (8, 12, 16, 18 or 20 positions).

(HKIWIUP) Our Price **$199**

KiWi+ Detent Discs
KiWi+ Detent Discs are available singly or in a set of four. In both cases you get to choose whichever discs you need.

(HDTDISC) Our Price **$24.95** each

(HDTDISC4) Our Price **$89** set of four

Choices include: 8, 12, 16 ,18, or 20 position detent disc

Landscape Bracket
The Landscape Bracket is a right angle bracket that allows you to mount the KiWi or KiWi+ upright camera bracket in a horizontal orientation. This bracket is primarily used for cameras that have a limited field of view and you need to limit the number of shots.

(HLDBRAC) Our Price **$42**

Flash Hotshoe Level
A dual-axis bubble level that slides into your camera's hotshoe. It's a useful tool to help level the camera on the upright camera bracket.

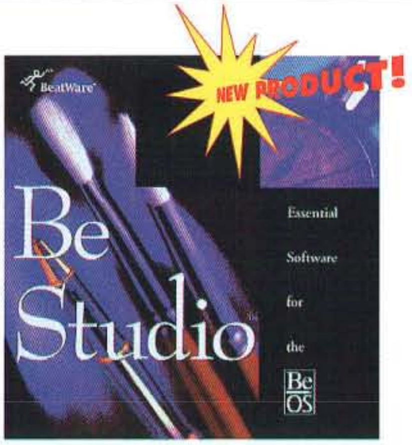
(HFLASH) Our Price **$39**

Offset Spacer
The Offset Spacer is a circular spacer that may be required for very narrow cameras (i.e. certain Ricoh digital cameras) in order to position the center of the lens over the pivot axis.

(HOFFSPAC) Our Price **$24.95**

Be Studio

by BeatWare



Developed specifically for the BeOS, Be Studio is the first graphics application to offer full multithreading for unbelievable responsiveness and unbeatable speed. Be Studio includes Paint for editing photos or creating original artwork and Draw, a vector-based tool for drawing crisp designs and prints. Be Studio offers the BeOS developer the ability to:

- Design application icons quickly and easily
- Edit and export images easily to the World Wide Web
- Manipulate several images at once without degrading performance
- View updates from a variety of perspectives simultaneously
- Import and Export images as GIF, JPEG, PNG, TIFF, and PNM files
- Plug-in third party tools and filters

Be Studio requires the BeOS, 16 MB of RAM (32 recommended), 3MB available hard disk space and a 256-color display adapter. Price includes all major and minor upgrades through version 2.0 via electronic distribution.

(SBESTUD) Our Price **$99**

Lowest price guaranteed. Feel Good about your purchase! 30-day return policy.

Order Toll-free 800-MACDEV-1 (800-622-3381)

Music Tracks

by BeachWare, Inc.

A new PC/Mac & Audio multimedia music CD-ROM. The clips include musical introductions, fanfares, background music, and more. This collection offers you 100 music clips stored in .WAV format for Windows, SoundEdit & AIFF formats for Macintosh and as Audio tracks for audio CS players. All of the music clips are completely license and royalty-free!! Mac System requirements: Mac Plus or greater, CD-ROM drive. PC system requirements: Windows 3.1 or later, Sound Blaster compatible board, CD-ROM drive.

(SMT) Our Price **$24.95**

MultiWare Multimedia Collection

by BeachWare, Inc.

Introducing a new Audio multimedia music CD-ROM for the Macintosh. This disc is a collection of clips ideal for Desktop Presentations and other Multimedia applications. This incredible collection of license-free media clips is bursting with 240+ color pictures and backdrops (PICT), 200+ sound & music clips (SoundEdit), 140+ QuickTime movies, and a variety of multimedia tools for use with the Macintosh.

(SMWMC) Our Price **$24.95**

webAlias 1.0

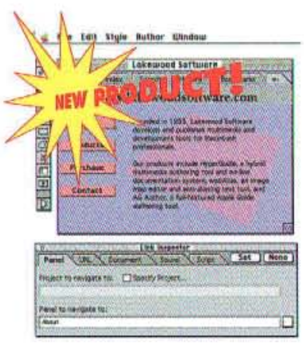
by Lakewood Software

webAlias 1.0 is an integrated image map editor and anti-aliasing text tool for web and graphic designers. Use webAlias to create complete web sites, single web pages, and graphic content for multimedia and web design projects. webAlias integrates support for line, shape, free form, field and button objects. webAlias' anti-aliasing features include support for embedded pictures and gradients in text, as well as multiple shadow and highlight effects.

(SWEBALS) Our Price **$129**

HyperGuide 1.0

by Lakewood Software



HyperGuide 1.0 is a hybrid multimedia authoring tool and on-line documentation system for the Macintosh and World Wide Web. HyperGuide provides integrated searching, indexing and bookmarking features. Supported media elements include: rectangle and scrolling fields, lines and shape fills, most QuickTime-supported image formats, anti-aliased text and QuickTime VR movies. HyperGuide also includes an integrated screen capture utility and user-configurable slide show mode.

(SHYPGUD) Our Price **$149**

WAIT... There's More!

Here are more products. For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us.

PRODUCT	CODE	OUR PRICE
AudioTrack	SAUDIOTRK	$270.00
Captivate 4.6 Essential Graphics Utilities	SCAPTIV	$79.00
Media Cleaner Pro	SMCPUP	$359.00

Power3D
by Techworks

The power of an arcade on your PowerPC Based on award winning 3Dfx VooDoo Graphics. The Power3D works with your existing graphics card and multi-sync monitor to provide you the absolute in 3D performance. Install the Power3D in your PowerPC (requires one available PCI slot in your system) and use the provided pass-through cable to turn your PowerPC into a Power Arcade system!
Power3D comes bundled with these awesome 3D enabled games:

- Quake®: Episode 1 (8 level) by Id Software
- MechWarrior® 2 by Activision
- VR Soccer™ by VR Sports (Actua™ for Europe)
- Weekend Warrior™ by Bungie

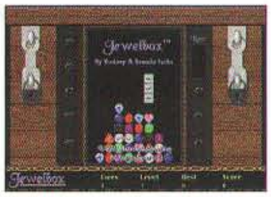
(SPWR3D) Our Price **$249**

Abuse
by Bungie Software

Abuse is 360 degrees of side-scrolling action. Run, jump, fall and fly in any direction - through industrial corridors, caverns and sewers. Destroy enemies in any direction with grenade launchers, rocket launchers, napalm and nova spheres! Avoid deadly traps with jet packs and turbo boost!
Key Features:

- Point and Kill Interface. Move and annihilate mutants in complete 360° freedom
- Blast your way through floors, walls and ceilings in search of the ultimate power-up!
- Abuse is 360 degrees of side-scrolling action. Run, jump, fall and fly in any direction – through industrial corridors, caverns and sewers

(SABUSE) Our Price **$51**



1000 Games for Macintosh
by BeachWare, Inc.

The best Macintosh game disc in the entire world, this CD-ROM contains over one thousand great shareware and public domain programs. Battle ugly aliens, blast apart run-away asteroids, deal yourself that royal flush or solve that 3-D puzzle, this disc has it all! System requirements: Mac Plus or greater, CD-ROM drive, and 2 MB of available RAM (4 MB of RAM when running under System 7).
(STGM) Our Price **$24**

Myth The Fallen Lords
by Bungie Software

***PC GAMES MAGAZINE'S*, Most Anticipated New Game Award**

The Fallen Lords is a fully 3D real-time strategy game of epic battle. A multimetric game, Myth: The Fallen Lords gives gamers unprecedented freedom to view their forces, orbiting around the heads of a formation or zooming in for a close-up on savage melee. Myth: The Fallen Lords includes maps designed for networking, and alternate networking scenarios like Assassin and King of the Hill.
(SMYTH) Our Price **$49**

Classic Arcade
by BeachWare, Inc.

Ten of your favorite coin-arcade games, redone with killer graphics and sounds! Walk through a virtual arcade and test your game playing skills with these exciting arcade classics. Ten games, including Moon Lander, Astro-Boing, Hyper Hockey, Ballistic Avenger, and more. System Requirements: Mac — Color Mac with 8 MB RAM, CD-ROM drive. PC 486 with 8 MB RAM, Sound card, SuperVGA, CD-ROM drive.
(SCLA) Our Price **$24**

Marathon Trilogy Box Set
by Bungie Software

The Marathon Trilogy Box Set brings all three Marathon games together in one affordable package, with tons of extras thrown in. Besides Marathon, Marathon 2: Durandal and Marathon Infinity, you'll also receive a staggering 1200 maps, featuring never-released Bungie maps and the winners of the Infinity Mapmaking Contest, The Marathon Scrapbook (a behind-the-scenes look at themaking of the Marathon games), Marathon collectables like the Marathon 3-sticker set, and to top it off, the award-winning game that laid the groundwork for Marathon: Bungie's breakthrough Pathways Into Darkness. The Marathon Trilogy Box Set is native to the Power Macintosh, utilizes the graphics acceleration of 630 and 6200 machines, is 8, 16 and 24-bit color capable, and can be played with joysticks and game pads. The package requires a 68040 or higher Macintosh, CD-ROM drive, 8-bit color monitor (13" recommended), and System 7 or later.
(SMTBS) Our Price **$65**

WAIT... There's More!

Here are more products. For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us.

PRODUCT	CODE	OUR PRICE
A Zillion Sounds	SAZS	$24.00
Casino!	SCAS	$24.00
Night Sky Interactive	SNSI	$24.00
Trivia Warehouse 2000	STW2K	$24.00

For Macintosh Programmers & Developers

MacTech® MAGAZINE

MacTech® Magazine

MacTech keeps Mac programmers & developers up to date with everything they need to know about software development. Topics like Rhapsody, Java, QuickTime, OPENSTEP, Objective-C, C/C++, Object Oriented Technologies, product reviews and much more!

Subscriptions:

(MTYRDM) US/Domestic for 12 issues **$47**
(MTYRCM) Canadian for 12 issues **$59**
(MTYRFM) International for 12 issues **$97**
Back Issues: each plus shipping (subject to availability) **$10**

MacTech® CD-ROM Volumes 1-12

- Includes Apple's *develop* issues 1-29 (1990-1997)
- Almost 1600 articles from all 139 issues of MacTech Magazine (1984-1996) and through may of 1997
- Improved hypertext, improved indices, and a new THINK Reference Viewer – for lightning quick access!
- New hyperlinks between articles
- 100+ MB of source code — use them in your applications, with no royalties!
- Full version of THINK Reference™ — the original online guide to Inside Macintosh, Vols. I-VI
- 80MB of FrameWorks/SFA archives and the most complete set of FrameWorks archives known
- Sprocket™! MacTech's tiny framework that compiles quickly and supports System 7.5 features
- The best threads from the Macintosh programmer newsgroups plus thousands of notes, tips, snippets, and gotchas
- Popular tools that Macintosh programmers use to increase their productivity and much more!

(SMTCD12) Volumes 1-12 Our Price **$129**

(SMTCD12U) Upgrade from any previous version Our Price **$49**

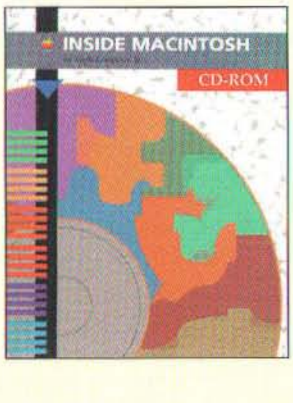
INSIDE MACINTOSH
by Apple Computer, Inc.

Inside Macintosh: CD-ROM

by Apple Computer, Inc.

More than 25 volumes in electronic form. Includes: QuickDraw™ GX Library, Macintosh Human Interface Guidelines, PowerPC System Software, Macintosh Toolbox Essentials and More Macintosh Toolbox, QuickTime and QuickTime Components. Access over 16,000 pages of information with Hypertext linking and extensive cross referencing.

(BIMCD) Our Price **$89**



Order Toll-free 800-MACDEV-1
(800-622-3381)

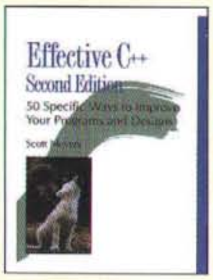
New Books!

The Official BBEdit Book

by Bob LeVitus and Natanya Pitts

The Official BBEdit Book makes it easy for today's Webmasters to speed their own interactive development projects using this powerful editing environment. Bare Bones incorporates features like floating palettes, HTML support, and syntax coloring to enhance an already extensive feature set, making BBEdit a leading Web authoring tool. Super-fast text processing, easy scripting, wide extensibility, strong GREP-style search-and-replace capabilities, and support for 13 languages make it easy to see why so many Web developers use BBEdit as their primary authoring tool.

(BOFBB) Our Price **$35**



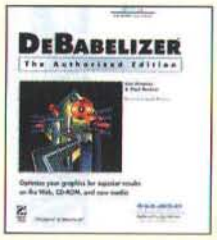
Effective C++, Second Edition: 50 Specific Ways to Improve Your Programs and Designs

by Scott Meyers

Effective C++, 2nd Edition includes: Expert guidance on object-oriented design, class design, and the proper use of inheritance

- An examination of the standard C++ library, including how the Standard Template Library and classes like string and vector affect the structure of well-written programs
- Discussions of late-breaking language features like in-class constant initializations, namespaces, and member templates
- Wisdom usually possessed by only the most experienced developers
- Effective C++ continues to be essential reading for every developer working with C++.

(BEFFC) Our Price **$34**



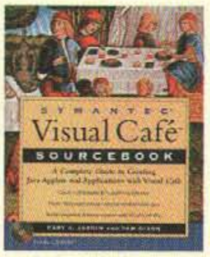
DeBabelizer

by Lise Despres and Paul Vachier

DeBabelizer: The Authorized Edition is the official guide for Web designers, multimedia creators, artists and production specialists who want to take advantage of this powerful tool.

- Create graphics and images for the Web that download fast and look amazing
- Optimize graphics for CD-ROM, video, and animation
- Optimize and manage colors using the SuperPalette(TM)
- Discover DeBabelizer tips and advice from industry experts
- Master basic manipulation techniques, including rotation, scaling, cropping, and text overlay
- Explore key production techniques in all areas of graphics processing

(BDEBTAE) Our Price **$40**

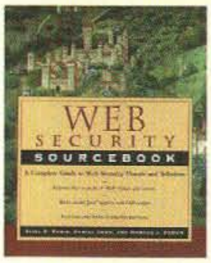


Symantec Visual Cafe Sourcebook

by Cary A. Jardin and Pam Dixon

Symantec Visual Cafe, the first visual Java development tool that gives programmers a sophisticated set of tools. This book teaches programmers how to use Symantec Visual Cafe to create Java applets. It provides a thorough introduction to the language and gives advanced Java programmers information on how to use Visual Cafe to create their own Java development tools.

(BSYMSOUR) Our Price **$35**



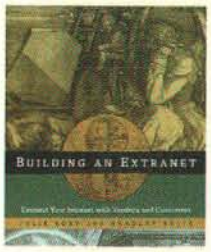
Web Security Sourcebook

by Aviel D. Rubin, Daniel Geer and Marcus J. Ranum

Technical tools and techniques for building secure web sites and applications

This book shows web masters, web managers, and web designers the hands on programming techniques necessary to build secure web sites. Readers will learn how to secure the server, use firewalls and cryptography, write secure Java applets and CGI scripts and more. Companion Web Site includes source code examples plus updates on the latest security threats and techniques.

(BWEBSER) Our Price **$26.99**

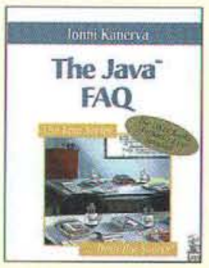


Building An Extranet: Connect Your Intranet With Vendors And Customers

by Julie Bort and Bradley Felix

Building an Extranet will help companies use their Intranet capabilities to supply information to selected customers and vendors through a secure Extranet. This book provides complete information and working details for building the network behind Intranets and Extranets, designing the applications, and getting everything up and running.

(BBLDEXT) Our Price **$26.99**



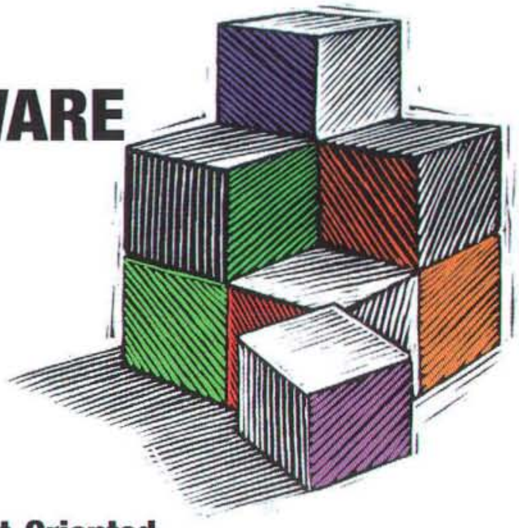
The Java FAQ

by Jonni Kanerva

Java FAQ provides an insider's view of the Java™ technology by posing and answering the most important, frequently asked questions about the Java programming language, Java applets, and Java stand-alone applications. The Java FAQ is unique in that it draws from the tens of thousands of questions sent to <java@java.sun.com> and provides authoritative answers direct from the creators of the Java programming language at JavaSoft.

(BJFAQ) Our Price **$25.13**

APPLE ENTERPRISE SOFTWARE



Getting Started With WebObjects
by Apple Enterprise Software

If you're a first time user, start here to learn the basics of how to create and run WebObjects applications.
(BGSWO) Our Price **$14**

WebObjects Developer's Guide
by Apple Enterprise Software

A guide to building and understanding WebObjects applications. Takes a close look at the WebObjects scripting language. Additional sections explain the WebObjects architecture and tells you how to integrate your code into the request-response loop, create reusable components, create client-side components, take advantage of powerful Foundation Framework features, and more. Filled with example code. For all WebObjects programmers.
(BWODG) Our Price **$16**

D'OLE Developer's Guide
by Apple Enterprise Software

Distributed OLE is available today, and this book shows you how to use it. Using real-world examples, the book steps you through the process of making your OPENSTEP objects available on Windows through OLE.
(BDOLEDG) Our Price **$22**

Discovering OPENSTEP, Mach
by Apple Enterprise Software

Introduces programmers to NeXT's OPENSTEP 4.0 Developer product by guiding them through the creation of three applications of increasing complexity. The tutorials demonstrate and explain programming techniques, Objective-C fundamentals, common APIs, and usage of the developement tools. Along the way they present summaries of important concepts and paradigms. The book also includes a chapter directing readers to programming resources, further information, and services such as training and support. An appendix offers a concise discussion of object-oriented programming.
(BDOSTEPM) Our Price **$15**

Discovering OPENSTEP, Windows
by Apple Enterprise Software

Discovering OPENSTEP provides an introduction to OPENSTEP programming on WIndows NT. It guides the reader through the creation of three applications of increasing complexity. Along the way, it explains concepts and illustrates aspects of Objective-C, OPENSTEP classes, the development environment, and programming techniques. A short appendix offers a summary of object-oriented programming.
(BDOSTEPW) Our Price **$16**

Object-Oriented Programming and Objective C
by Apple Enterprise Software.

An introduction to the principles of object-oriented programming in OPENSTEP and the official description of the Objective-C language. Objective-C is easy to learn and use because it adds very little syntax to the C programming language. It's dynamic nature allows you to accomplish things not possible in most other object-oriented languages. For any OPENSTEP programmer.
(BOOPOC) Our Price **$24**

Working w/ Interface Builder (for EOF)
by Apple Enterprise Software

A hands-on, award-winning book designed to help you get your job done with the updated Interface Builder, released with NEXTSTEP 3.3 and the Enterprise Objects Framework 1.1. For any programmer using Interface Builder to design objects that truly work in NEXTSTEP.
(BWIB) Our Price **$24**

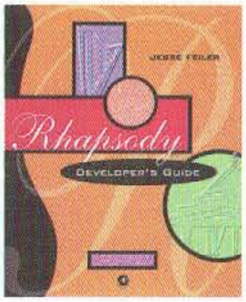
Using EOF 2.1 w/ OPENSTEP (Mach & Windows)
by Apple Enterprise Software

Using Enterprise Objects Framework with OPENSTEP describes how to create an Enterprise Objects Framework application on OPENSTEP. It includes a tutorial and a chapter on creating a user interface for an OPENSTEP Enterprise Objects Framework application.
(BUEOFO) Our Price **$14**

EOF Developer's Guide for EOF 2.1 (Mach & Windows)
by Apple Enterprise Software

The Enterprise Objects Framework Developer's Guide describes how to develop database applications using the Enterprise Objects Framework tools and classes. It includes an architectural overview of the product, and descriptions of programming tips and techniques. An appendix offers a summary of Entity-Relationship Modeling.
(BEOFDG) Our Price **$24**

EOF Developer's Guide for EOF 2.0	(BEOFDG20)	Our Price	**$24**
EOF Developer's Guide for EOF 1.x	(BEOFDG1X)	Our Price	**$24**

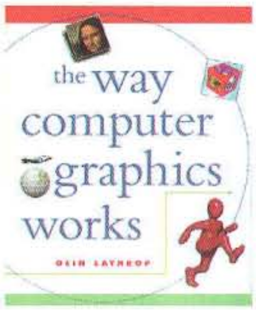


Rhapsody Developer's Guide

by Jesse Feiler

Covers the basic architectural principles of Rhapsody: the Mach microkernel, object-oriented programming, and the elements of a modern OS such as preemptive multitasking, protected memory, and symmetric multiprocessing. Also shows ways of getting to this new environment — Objective C, conversion tools, and the integration of Java — to develop Rhapsody products. Paperback, 450 pages.

(BRDG) Our Price **$35**



The Way Computer Graphics Works

by Olin Lathrop

A complete guide to mastering computer graphic basics. It is written in a frank, down-to-earth style covering everything from how computer graphics are different from fine art and photographs, to modeling, pixels, and the principles of animation. All of this is done without resorting to mind-numbing equations and impenetrable technical jargon.

(BWCGW) Our Price **$29.65**

Debugging Macintosh Software with MacsBug

Includes CD-ROM!

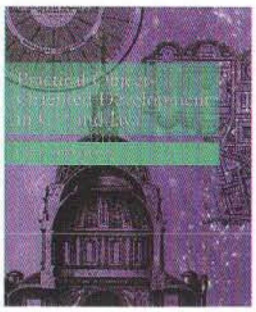
by Konstantin Othmer and Jim Straus

MacsBug, from Apple Computer, Inc., is the leading debugging software program for the Macintosh. This book/disk package is an all-in-one kit for using MacsBug. Chapter 1 introduces MacsBug and describes the contents of the rest of the book. Chapter 2 describes how to install MacsBug and enough low level details about the Macintosh so that you can use MacsBug. Includes MacsBug 6.2 on disk.

(BDMSWM) Our Price **$31**

Order Toll-free
800-MACDEV-1
(800-622-3381)

Check out our Web site!
• Full product descriptions • Hundreds of more products
http://www.devdepot.com



Practical Object-Oriented Development in C++ and Java

by Cay S. Horstmann

This book offers advice on real-world ways to use these powerful programming languages and techniques. Using the Unified Modeling Language (UML) methodology, expert Cay S. Horstmann gives you clear, concise explanations of object-oriented design, C++, and Java in a way that makes these potentially daunting operations more accessible than they've ever been before.

(BPOOD) Our Price **$31**

WebMaster in a Nutshell, Deluxe Edition

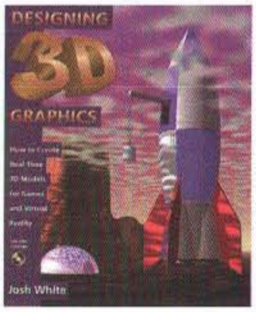
by O'Reilly & Associates, Inc.

Includes CD-ROM!

Cross-platform, completely portable, and lightning fast, the CD-ROM is an invaluable addition to the webmaster's toolbox. The CD-ROM contains the Web Developer's Library — the full text of the latest editions of five popular O'Reilly titles: "HTML: The Definitive Guide, 2nd Edition"; "JavaScript: The Definitive Guide, 2nd Edition"; "CGI Programming on the World Wide Web"; "Programming Perl, 2nd Edition"; and "WebMaster in a Nutshell." The Deluxe Edition also includes a printed copy of "WebMaster in a Nutshell," the all-inclusive quick reference that belongs next to every webmaster's terminal. Includes CD-ROM & 356 page book.

Requirements: The CD-ROM is readable on all platforms, but requires a web browser that supports HTML 3.2, Java, and JavaScript.

(BWMNUTD) Our Price **$62**



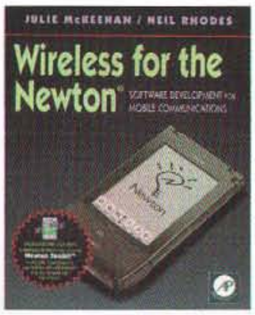
Designing 3D Graphics

Includes CD-ROM!

by Josh White

In this powerful book/CD-ROM package, top computer graphics artist Josh White tells you everything you need to know to create sophisticated real-time 3D graphics for computer games and virtual reality. This book contains the in-depth knowledge of software tools and hands-on modeling techniques that Josh White has learned while creating artwork for over 20 commercial games, including Descent, Zone Raiders, Locus, Legoland, and others.

(BD3DG) Our Price **$35**



Wireless For The Newton

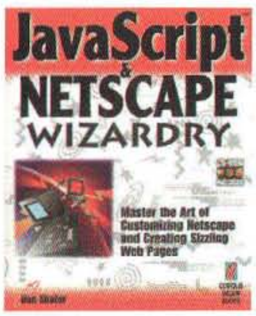
Includes Disk!

by Julie McKeehan and Neil Rhodes

A book that picks up where Programming for the Newton left off, teaching the reader how to develop Newton software on the Macintosh. The enclosed floppy disk provides a sample application, as well as a fully functional demonstration version of Newton Toolkit.

- Learn to develop Newton software on the Macintosh
- Hands-on Newton environment training with sample code
- Includes disk with sample source code for a Newton application, as well as demonstration NTK – the complete development environment for the Newton

(BWIRELESS) Our Price **$31**



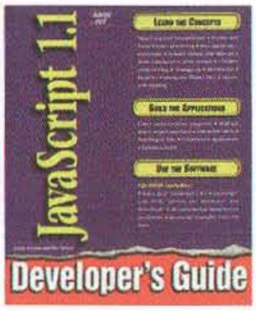
JavaScript & Netscape Wizardry

by Dan Shafer

Includes CD-ROM!

The perfect book to show you how to turn Netscape into your own personal, customized operating system. Provides the inside tips and techniques for making your Web pages much more attractive. Shows you how to use all of the key features of the JavaScript language, including objects, methods, properties, events, and much more. Includes CD-ROM with numerous interactive scripts written in JavaScript you can add to your Web pages today. A complete set of the best Java applets. Useful plug-ins designed to supercharge Netscape and resources to help JavaScript programmers.

(BJNWIZ) Our Price **$31**



JavaScript 1.1 Developer's Guide

by Arman Danesh and Wes Tatters

Includes CD-ROM!

Written by developers for developers. An advanced guide to creating professional Web applications with JavaScript 1.1 as deployed in Netscape Navigator 3.0, Microsoft Internet Explorer 3.0, and LiveWire. Includes CD-ROM with Sun's Java Developer's Kit, JavaScript and HTML Editors for Windows and Macintosh, 20 contributed ready-to-run JavaScripts and JavaScript examples from the book.

(BJSDG) Our Price **$44**

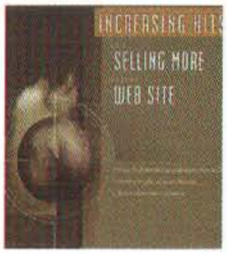
Linux Configuration & Installation, 3rd Edition

by Patrick Volkering, Kevin Reichard, and Eric F. Johnson

Includes CD-ROM!

Linux, the leading UNIX variant, has garnered loads of attention within the UNIX community. The amazing thing about Linux is that you don't need a workstation to run it. Linux Configuration & Installation, Second Edition lets you run Linux today. Program with Linux using C, C++, Perl, and Tcl/Tk. The 2 CD-ROM pack offers one of the most popular Linux distributions, Slackware 96, and comes directly from Patrick Volkering, the creator of Slackware.

(BLCI2) Our Price **$35**

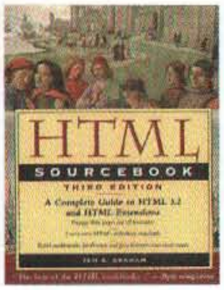


Increasing Hits and Selling More on your Web Site

by Greg Helmstetter

Written especially for entrepreneurs, corporate marketing managers, small business owners, and consultants, this valuable guide gives you rare tips and tricks you need to know to make your site a commercial success.

(BIHSMWS) Our Price **$22.45**



HTML Sourcebook, 3rd Edition

by Ian S. Graham

Critics everywhere agree, HTML Sourcebook is the best guide to HTML for Web professionals. That's because no other book makes it so easy for you to quickly master all the commands, tools, and expert techniques you need to create cutting-edge Web page documents. Completely revised and expanded by nearly 50 percent, this new third edition of the best-selling guide to HTML gives you the complete lowdown on all the changes and enhancements to the HTML, HTTP, and URL standards.

(BHTMLS) Our Price **$26.95**

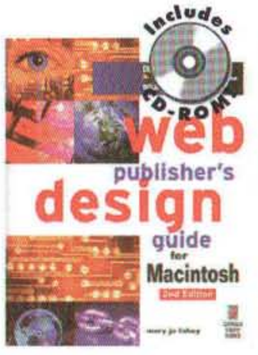
Teach Yourself Java for Macintosh in 21 Days

Includes CD-ROM!

by Laura Lemay and Charles L. Perkins with Timothy Webster

Add interactivity and multimedia to Web pages! A step-by-step guide to make your Website come alive. Learn the basics of programming Java applets and the concepts behind the Java language. Includes CD-ROM with a limited version of Roaster, the first commercial, integrated applet development environment for Java for the Macintosh!

(BJAVAMAC) Our Price **$36**

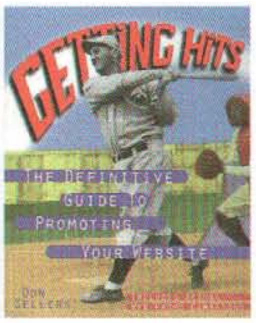


Web Publisher's Design Guide for Macintosh, 2nd Edition

by Mary Jo Fahey

This is the only book that takes you step-by-step through real projects designed by talented new media artists. Internet design experts share their design secrets and art files (look for art files on the companion CD-ROM by artist's last name). This expanded new edition includes Photoshop, Illustrator and DeBabelizer tricks from the first edition plus countless new ways to liven up your Web pages with 3D graphics, sound, movies, and much more.

(BWPDG2) Our Price **$35**

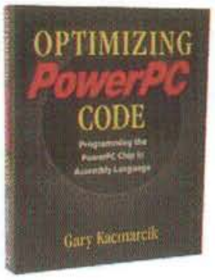


Getting Hits–The Definitive Guide To Promoting Your Website

by Don Sellers

Getting Hits explains in easy-to-understand language the underlying concepts behind the art of Web site promotion. Just a few of the topics you'll learn include: using search engines with URL's; finding related Internet groups or lists; understanding the nuances of click throughs and ad rates; and distributing press releases to key Internet contacts. With this book, you'll go beyond the conceptual and actually follow real-world tested promotional campaign strategies. Bring the world to your Web site!

(BGHITS) Our Price **$17.95**

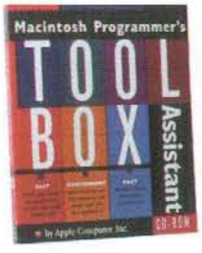


Optimizing PowerPC Code: Programming the PowerPC in Assembly Language

by Gary Kacmarcik

Take full advantage of the potential of the PowerPC by mastering the Assembly Language techniques. Learn to produce faster more robust software!

(BOPTPPC) Our Price **$35**



Programmer's Toolbox Assistant CD-ROM

Instant electronic access to Inside Macintosh essentials.
by Addison-Wesley Publishing

Get quick access to reference pages for over 4,000 Toolbox calls in your system software from their development environment. Essential information for Macintosh software developers. Hypertext links allow programmers to view related topics easily. The ultimate electronic reference tool for Macintosh programmers.

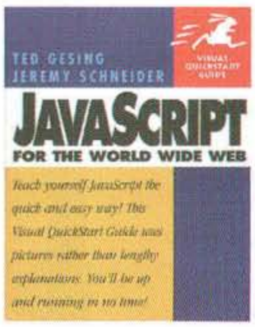
(STBASST) Our Price **$89**

HTML For The World Wide Web, 2nd Edition

by Elizabeth Castro

Teach yourself Hypertext Markup Language the quick and easy way! This Visual QuickStart Guide uses pictures rather than lengthy explanations. You'll be up and running in no time. If you need to learn HTML fast – this is book is for you.

(BHTMLW2) Our Price **$16.15**

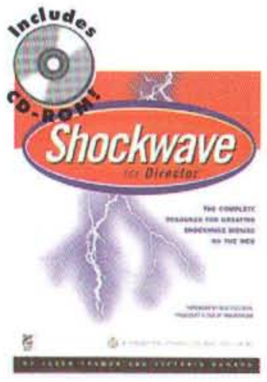


JavaScript For The World Wide Web

by Ted Gesing and Jeremy Schneider

This book takes an easy, visual approach to teaching JavaScript, where pictures guide you through the software and show you what to do. Works like a reference book, you look up what you need and then get straight to work. No long winding passages, concise, straightforward commentary explains what you need to know.

(BJWWW) Our Price **$16.15**



Macromedia Shockwave for Director

by Jason Yeaman and Victoria Dawson

The complete resource for creating Shockwave movies on the Web. This hands-on reference makes it easy to create Shockwave movies and put them on the Web. Expert tips from the creators of Macromedia's first Shockwave movies, together with detailed examples and instruction, provide everything you need to get started. Includes CD-ROM.

(BMSFD) Our Price **$27**

Lowest price guaranteed. Feel Good about your purchase! 30-day return policy.

Check out our Web site!
• Full product descriptions • Hundreds of more products
http://www.devdepot.com

Metrowerks CodeWarrior Programming

by Dan Parks Sydow

Includes CD-ROM!

Includes CodeWarrior Lite, and Full Coverage of PowerPlant™. The best information on Metrowerks CodeWarrior, giving full coverage to the Gold Edition. CD includes Code Warrior Lite.

(BCWPROG) Our Price **$35**

CodeWarrior Software Development Using PowerPlant

by Jan L. Harrington

Includes CD-ROM!

C++ programmers will learn to develop object-oriented software applications for the Mac and Power Mac using the PowerPlant environment and the classes that support it. Covers CodeWarrior 8. Included CD-ROM contains source code for all the programming examples in the book and Metrowerks CodeWarrior Lite.

(BCWSWDEV) Our Price **$31**

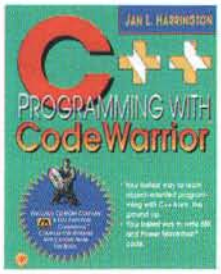
Inside PowerPlant

by Metrowerks

Create PowerPlant applications using the CodeWarrior IDE and PowerPlant Constructor. Full descriptions of major PowerPlant classes and resources. Included are the PowerPlant Constructor Manual, including View, TextTraits and Custom Types editing, and PowerPlant Library Reference, covering all classes and functions in PowerPlant.

(BINSPP) Our Price **$34**

SEE RELATED CATEGORY: Dev. Environment



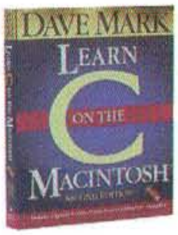
C++ Programming with CodeWarrior

by Jan L. Harrington

Includes CD-ROM!

Beginning OOP for the Macintosh and Power Macintosh and Mac OS compatibles. Learn object-oriented programming techniques using C++ as the example language and Metrowerks and CodeWarrior as the example compiler. Enclosed CD contains example code from the book and a full-function Metrowerks CodeWarrior.

(BCPPCW) Our Price **$33**



Learn C on The Macintosh Second Edition

by Dave Mark

Includes CD-ROM!

New revised edition! Easy-to-understand – everything you need to start programming. Updated and enhanced exercises that lead you step by step. You'll learn function, variables, point datatypes, data structures, file input and output and more! Includes CD-ROM with Metrowerks CodeWarrior™ Lite.

(BLEARNC2) Our Price **$33**

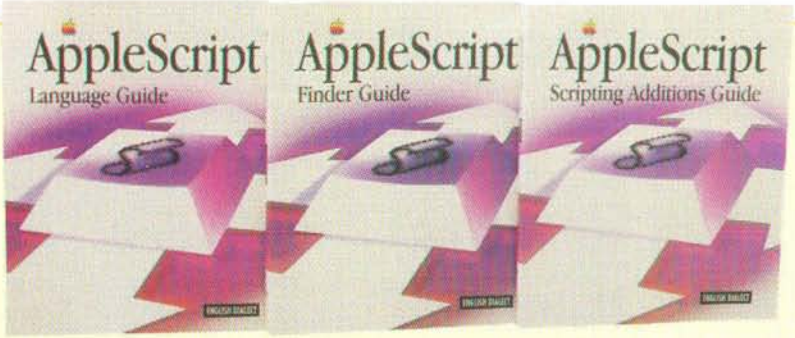
Order Toll-free
800-MACDEV-1
(800-622-3381)

MacTech® Magazine

MacTech keeps Mac programmers & developers up to date with everything they need to know about software development. Topics like Rhapsody, Java, QuickTime, OPENSTEP, Objective-C, C/C++, Object Oriented Technologies, product reviews and much more!

Subscriptions:

(MTYRDM) US/Domestic for 12 issues **$47**
(MTYRCM) Canadian for 12 issues **$59**
(MTYRFM) International for 12 issues **$97**
Back Issues: each plus shipping (subject to availability) **$10**



AppleScript Language Guide

by Apple Computer, Inc.

A complete reference for anyone using AppleScript to modify existing scripts or to write new ones. Contains useful information for programmers who are working on scriptable applications or complex scripts. Features detailed definitions of AppleScript terminology and syntax in the following categories: Value classes, commands, objects and references to objects, expressions, control statements, handlers, and script objects. Includes many sample scripts, discusses advanced topics such as writing command handlers for script applications, the scope of script variables and properties declared at different levels in a script, and inheritance and delegation among script objects.

(BALG) Our Price **$26.95**

SEE RELATED CATEGORY: Scripting

AppleScript Applications: Building Applications with FaceSpan and AppleScript

Includes CD-ROM!

by John Schettino Affiliation & Liz O'Hara

Build complete AppleScript applications using FaceSpan, a user interface development tool that makes AppleScript applications truly "Mac-Like". Uses a step-by-step approach to demonstrate techniques for building applications through illustrations and samples. Provides Graphical User Interface (GUI) design tips and practical approaches for implementation. Contains one CD-Rom with AppleScript 1.1, a demonstrations version of FaceSpan 2.1, source code for all example applications numerous AppleScript shareware and demonstrations programs. Contains a section on debugging AppleScript applications using FaceSpan.

(BAPSCAP) Our Price **$31**

Special Edition Using CGI, 2nd Edition

Includes CD-ROM!

by Jeffry Dwight, Michael Erwin and Robert Niles

This complete reference provides professional Web developers and advanced personal users with the latest information on using CGI (Common Gateway Interface) to interact with databases.

- Explains client and server uses of CGI
- Provides extensive coverage of live audio and video feeds, user chat and interaction, and CGI security
- Features separate chapters devoted to language-specific tips, tricks, and traps
- CD ROM is loaded with the HTML and CGI sample code from the book
- Includes applications for guest books, mail and new gateways, browser identification, access restriction, and shopping carts

(BSEUCGI) Our Price **$44**

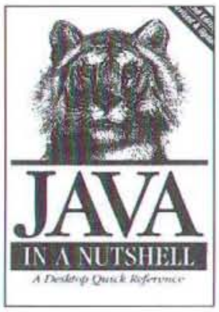
JavaScript for the Macintosh

by Matt Shobe and Tim Ritchey

Includes CD-ROM!

Allows non-programmers to take advantage of the power of Netscape Navigator. Expand the capabilities of your Web page, without having to understand C or C++. CD-ROM contains "Wizlets" that allows you to easily create your own JavaScripts. Takes you step-by-step through programming cross-platform JavaScripts. Details how to create JavaScripts for JavaScript-aware Web browsers.

(BJAVASCRPTJ) Our Price **$40**



Java in a Nutshell, 2nd Edition

by David Flanagan

A detailed overview of all of the new features in Java 1.1, both on a package-by-package basis and in terms of overall functionality. A comprehensive tutorial on "inner classes" that explains how to use all of the new types of inner classes: static member classes, member classes, local classes, and anonymous classes. Practical, real-world example programs that demonstrate the new features in Java 1.1, including object serialization, the new AWT event handling model, internationalization, and a sample Java Bean.

(BJNUT2) Our Price **$17.95**

AppleScript Finder Guide, English Dialect

by Apple Computer, Inc.

Provides definitions for Finder object classes and commands. Write, record, or run scripts that trigger the same desktop actions that you trigger using the keyboard and mouse.

(BAFG) Our Price **$17.95**

SEE RELATED CATEGORY: Scripting

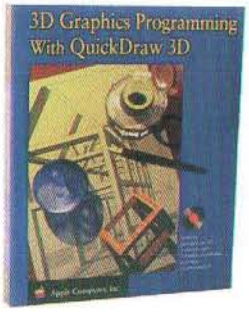
Inside CodeWarrior Professional

by Metrowerks

Includes CodeWarrior IDE User's Guide. This is the printed version of the documentation provided on the CD. Covers CodeWarrior Professional Release, the debugger and associated tools.

(BINSCWP) Our Price **$34**

SEE RELATED CATEGORY: Dev. Environment



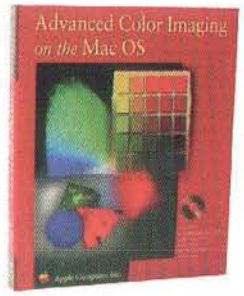
3D Graphics Programming Using QuickDraw 3D

Includes CD-ROM!

by Apple Computer, Inc.

Incorporate spectacular 3D graphics into your applications. Explore QuickDraw 3D, a revolutionary graphics extension to the Mac OS for Power Macintoshes. CD contains the complete QuickDraw 3D system itself and a complete database of the QuickDraw 3D API, allowing you instant access to the hundreds of graphics calls via a fast viewing engine. Book/CD-ROM, 640 pages.

(B3DGRAP) Our Price **$35**



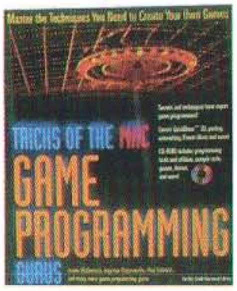
Advanced Color Imaging on the Mac OS

Includes CD-ROM!

by Apple Computer, Inc.

Enhance your software's color capabilities with step-by-step instructions. Augment the color support supplied with QuickDraw, and QuickDraw GX. Use the Pallette Manager to get the best colors on limited displays. Match colors between screens and input/output devices (scanners & printers). CD includes a complete reference information in both QuickView and Acrobat formats. Plus, a sample application demonstrating ColorSync programming techniques.

(BADVCI) Our Price **$33**

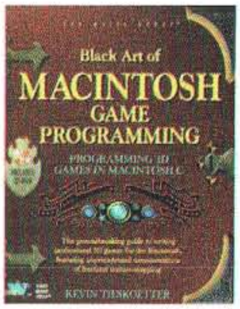


Tricks of The Mac Game Programming Gurus

by McCornack, Ragnemalm, Celestin, et al.

For beginning to expert game programmers. Complete overview of all the necessary components of game programming on the Macintosh. Packed with valuable tools, utilities, sample code, CodeWarrior™ Lite and game demos. QuickDraw 3D and Power Mac optimization and inside info on how Glypha III was created. Hundreds of tried-and-true tricks, tips, and insider secrets from well-known Mac game programming experts.

(BTRICKS) Our Price **$45**



Black Art of Macintosh Game Programming

Includes CD-ROM!

by Kevin Tieskoetter

Develop your own 3D games in C on the Mac. Includes CD with project files for both Symantec C and Code Warrior. Create freeform texture-mapped games and polygon graphics. Control dynamic source code — all compatible as native to the Power Mac. Write directly to the screen, bypassing QuickDraw.

(BBLACK) Our Price **$35**

WAIT... There's More!

Here are more products. For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us. Our prices on books are at least 10% off list price.

PRODUCT	CODE	PRICE
Development Environments		
AppleScript Applications: Building Applications w/FaceSpan	BAPSCAP	$31.00
C++ Programming	BCPPMACAP	31.00
C/C++ SDK User's Guide	BCPPUSER	29.00
CodeWarrior Inside PowerPlant	BINSPP	34.00
Cyberdog Programmers Kit	BCYBERDOG	34.00
Dan Shafer Presents the Power of Prograph CPX	BDANPRO	19.95
Inside CodeWarrior Book	BINSCW	34.00
Instant CORBA	BIC	17.99
Last Resort Programmers Edition	BLSTRSRT	74.00
Macintosh C Programming Primer Volume I	BCPRIM1	24.25
Macintosh C Programming Primer Volume II	BCPRIM2	24.25
Macintosh Pascal Programming Primer Volume I	BPASCPRI	24.25
Mastering the Think Class Library	BMASTERTCL	26.95
Mastering the Toolbox Using THINK C	BCPRIM2	24.25
Objective-C - Object Oriented Programming	BOBJCOOPT	34.00
Presenting Magic Cap	BPRESMAGIC	15.25
Real World Apple Guide	BREALWLD	35.00
Symantec C++ Programming	BSYMCPP	39.00
Taligent's Guide to Designing Programs	BTALIGENT	17.55
The Power of Prograph CPX	BDANPRO	19.95
Visual Programming with Prograph CPX	BVISPRO	30.00
Wireless For The Newton	BWIRELESS	31.00
Hardware		
Apple CD-ROM Handbook	BCDHAND	14.36
Designing Cards & Drivers for the Macintosh	BCARD	26.96
LaserWriter Reference	BLASERREF	17.96
PCI System Architecture 3rd Edition	BPCISYS	31.00
PowerPC System Architecture	BPPCARCH	31.00
Internet Related		
1994 Internet White Pages	B94WHITE	26.95
Active Java	BACTJAVA	23.36

America Online for Dummies	BAOLDUM	17.95
CGI By Example	BCGIBE	31.00
Building & Maintaining an Intranet w/ the Macintosh	BBAMAI	45.00
Claris Home Page Companion	BCHPC	26.95
Computer Privacy Handbook	BPRIV	22.45
Developing CGI Applications with Perl	BDCGIA	26.95
E-Mail Essentials	BEMAILE	22.45
Elements of E-Mail Style	BEMAIL	13.45
Hooked on Java	BHJAVA	26.95
Instant Internet Guide	BINSTANT	13.45
Internet Book	BTHENET	22.50
Internet for Dummies 2nd Edition	BNETDUM2	17.99
Internet for Dummies Quick Reference	BDUMQCK	8.05
Internet for Macs for Dummies	BNETDUM	17.95
Internet for Macs for Dummies Bestseller Edition	BIFMFDBE	35.00
Internet Power Tools	BPWRTOOL	36.00
Internet Publishing with Adobe Acrobat	BIPWAA	36.00
Internet, The, Deluxe Edition	BNETDELUX	31.00
Intranet Web Dev.: Enterprise Alternatives to Client/Server	BINTWD	44.00
Java Essentials for C/C++ Programmers	BJAVAESSEN	17.95
Java in a Nutshell	BJAVANUT	13.45
Java Language API SuperBible	BJLAS	53.00
Java Programming with CORBA	BJPWC	26.99
JavaScript Cookbook	BJSCB	44.00
JavaScript for Macintosh	BJAVASCRPT	40.00
Learn HTML on the Macintosh	BLHTML	26.95
Learn Java on the Macintosh	BLJAVA	31.00
Mastering Netscape 2.0 for Macintosh, Second Edition	BMASNET2	36.00
Measuring the Impact of Your Web Site	BMIYWS	26.95
More Internet for Dummies Starter Kit	BDUMNET	17.95
Mosaic for Dummies	BMOSDUM	17.99
Net Chat	BNETCHAT	17.00
NetObjects Fusion Handbook	BNETOFH	45.00
Perl Quick Reference	BPERLREF	17.99
Planning and Managing Websites	BPLANWEB	35.00
Protect Your Privacy on the Internet	BPYP	26.99
Providing Internet Services via the MacOS	BPROVNET	31.00
Publish it on the Web	BWEBPUB	31.00
TCP/IP Vol 1-Vol 2 Bundle	BTCP12BNDL	99.00
Teach Yourself Java in 21 days	BJAVAMAC	36.00
The Internet Marketing Plan	BTIMP	35.00
The Internet Strategic Plan	BTISP	22.45
The Web Navigator	BTWN	22.45
Underground Guide to Telecommuting	BUNDER	22.45
Using Lotus Notes as an Intranet	BULN	40.00
Web Graphics Tools and Techniques	BWGTT	35.00
Web Head Mac Guide	BWEBHEAD	22.45
Web Marketing Cookbook	BWMCB	35.00
Web Page Scripting Techniques	BWEBPST	45.00
Web Publishing with Adobe Acrobat and PDF	BWPWAA	35.95
Web Weaving	BWWEAV	22.45
Webmaster Macintosh	BWEBMAS	26.95

Scripting and Solutions

AppleScript Applications: Building Apps with FaceSpan	BAPSCAP	31.00
AppleScript Scripting Additions Guide	BSCRADD	17.05
Applied Macintosh Scripting	BAPPLIED	31.00
Complete HyperCard 2.2 Handbook	BHYPCRD2	31.00
Complete HyperTalk 2.2	BHYPCRD2	31.00
Danny Goodman's Apple Guide Starter Kit	BDGAGSK	31.00
HyperCard Stack Design	BHYPSTA	19.95
JavaScript for Macintosh	BJAVASCRPT	40.00
Perl Quick Reference	BPERLREF	17.99
Real World Apple Guide	BREALWLD	35.00

Technical Reference

Active Java	BACTJAVA	23.36
Apple CD-ROM Handbook	BCDHAND	14.36
Art of Human Interface Design	BAHID	29.65
Black Art of Mac Game Programming	BBLACK	35.00
C++ for Dummies	BCPPDUM	22.46
C for Dummies Vol. 1	BCDUM	17.95
Developing Object Oriented Software for the Macintosh	BDEVOB	26.06
Extending the Mac Toolbox	BETMT	22.46
Foundations of Macintosh Programming	BFOUND	35.00
Fragment of Your Imagination	BFRAG	35.00
Guide to Macintosh Software Localization	BLOCALIZ	24.26
Guide to Macintosh System 7.5	BSYS7.5	22.50
Inside AppleTalk	BAPTALK	31.00
Inside the Macintosh Communications Toolbox	BCOMM	22.45
LaserWriter Reference	BLASERREF	17.96
Learn C++ on the Macintosh	BLRNCPP	35.00
Learn C on the Macintosh, 1st Edition	BLEARNC1	31.00
Learn C on the Macintosh, 2nd Edition	BLEARNC2	33.00
Mac Programming for Dummies	BMACDUM	17.95
Macintosh C Programmer Primer Volume 1	BCPRIM1	24.25
Macintosh C Programmer Primer Volume 2	BCPRIM2	24.25
Macintosh OLE2 Prog. Reference Working with Objects	BOLE2	40.00
Macintosh Pascal Programming Primer Volume I	BPASCPRI	24.25
Macintosh Programming Secrets 2nd edition	BPSECRET	28.76
Macintosh Programming Techniques	BPTECH	31.00
Microsoft Visual J++ 1.1 Sourcebook	BMVJS	35.00
More Mac Programming Techniques	BMORETECH	34.00
Network Frontiers Bundle	BNETFB	59.00
Newton Programming Guide	BNEWTPGUID	40.00
Object Oriented Programming Design	BOOPRODES	20.66
Optimizing PowerPC Code	BOPTPPC	35.00
Perl Quick Reference	BPERLREF	17.99
PostScript Language Reference	BPSLANREF	29.66
Powerbook: Digital Nomad's Guide	BPBTDNG	22.46
PowerPC Programmer's Toolkit	SPPCPT	40.00
Programming Introduction to the Macintosh Family	BFAMILY	22.46
Programming for System 7	BSYS7	24.25
Programming with AppleTalk	BPROAT	22.45
QuickTime - Official Guide for Macintosh Users	BQTGUIDE	45.00
Real World Apple Guide	BREALWLD	35.00
ResEdit All Night Diner	BRESDINE	22.45
ResEdit Complete, 2nd Edition	BRESED2	31.00
ResEdit Reference	BRESEDREF	26.96
Software by Design: Creating User Friendly Software	BDESIGN	26.95
Symantec C++ for the Macintosh: The Basics	BSCFTMTB	31.00
Teach Yourself Macintosh C++ in 21 Days	BCPP21D	26.99
Technical Introduction to the Macintosh Family	BTITTMF	24.26
Tog on Software Design	BTOG	26.95
Writing Localizable Software	BLOCAL	24.25

Miscellaneous

3D Graphics-Tips, Tricks, & Techniques	B3DGTTT	31.00
A Fragment of Your Imagination	BFRAG	35.00
Adobe Premiere for the Macintosh	BPREM	44.00
America Online for Dummies	BAOLDUM	17.95
AppleGuide Complete	BAPLGD	35.00
Art of Human Interface Design	BAHID	35.00
CD-ROM Guide to Multimedia Authoring	BCDMULTI	40.00
CompuServe for Dummies	BCSDUM	17.95
Cyberpunk Handbook	BCYBPUNK	8.95
Danny Goodman's Apple Guide Starter Kit	BDGAGSK	31.00
Danny Goodman's Macintosh Handbook	BGOODHB	26.95
FrameWorks Source Code Disk	MTFWSC	9.95
FrameWorks Magazine Back Issue	MTFWBACK	8.00
Graphic Gems 2	BGEMS2	44.00
Graphic Gems 4	BGEMS4	44.00
Graphic Gems V	BGEMS5	44.00
Infini-D Revealed	BINFDREV	40.00
Inside Director 5 with Lingo for Macintosh	BID5WLM	44.00
Late Night with MacHack	BLATE	26.95
Mac Bathroom Reader	BBATH	11.70
Macromedia Director Lingo Workshop, 2nd Edition	BMDLW2	40.00
Mac Screamer: The Ultimate Macintosh Supercharging Kit	BSCREAM	31.00
Macintosh Crash Course	BCRASH	26.95
MacTech Back Issues	MTBACKISS	10.00
Macworld Ultimate Macintosh Programming Book	BULTMAC	35.00
MADACON '93 CD-ROM	SMADA93	9.95
Multimedia Authoring: Building and Developing Documents	BMMAUTH	31.00
Multimedia Starter Kit for Macintosh	BMMSTART	27.00
Profit from Experience	BPROFIT	22.45
ResEdit Complete, Second Edition	BRESED2	31.00
Sad Macs, Bombs and Disasters	BSADMAC	22.45
Standards For Online Communication	BSFOC	40.00
The Elements of E-Mail Style	BEMAIL	13.45
The Software Developer's & Marketer's Legal Companion	BSDAMLC	33.00
Tog on Software Design	BTOG	26.95
Tricks of the Mac Game Gurus	BTRICKS	45.00
Zen and the Art of Resource Editing	BZAORE	27.00

All entries in this index are alphabetized. For your convenience the product names are **bold**, book names are *italicized* and company names are in plain text.

The NEW BBEdit 4.5!

It still Doesn't Suck.®

There has never been a better time to buy BBEdit.

Upgrade from BBEdit Lite, Codewarrior, MPW, Symantec C++ or a number of other products for just $79.

This offer is available direct from Bare Bones Software, Inc. To order, call us at (617) 778-3100 or visit us online.

BBEdit User Manual
Bare Bones Software, Inc.
BBEdit
High Performance Text & HTML Editing
BBEdit 4.5

<http://www.barebones.com/>

Bare Bones Software, Inc.
P.O. Box 1048, Bedford, MA 01730 • main 617-778-3100 • fax 617-778-3111

CodeWarrior Professional Release 2. Just out and ready for some serious, industrial-strength programming. Thi is the only Integrated Development Environment that allows you to edit, compile and debug C, C++, Java and Pascal progran for multiple target processors and operating systems. We're still the codewriter's best friend and now we're better tha ever. Here's why:

- ✗ portable project files — may be used interchangeably between MAC and Windows platforms
- ✗ file compare and merge — select two files to graphically view differences and merge changes
- ✗ browse across subprojects — source code browser works across targets and subprojects

You can count on CodeWarrior Professional Release 2. We're powerful, proven, easy-to-handle and we've had all our shots.

CodeWarrior.
Mark your territory.

metrowerks® www.metrowerks.cor

Come sniff around our specials
at Developer Central at MAC World, San Francisco.